Planificación, desarrollo e integración de la infraestructura tecnológica de gestión de una PYME con plataformas de código abierto.

Trabajo Fin de Grado.

Grado en Ingeniería Telemática.

Autor: Enrique Guadalupe Estévez

Director: José Antonio Díaz Nicolás (Sugerendo Sistemas)

Tutor: Dr. Jesús Arias Fisteus

Leganés, Septiembre 2014

Agradecimientos

A mis padres, Pablo y Luisa que desde pequeño me han ayudado a estudiar lo que he querido. Sin ellos no hubiera podido culminar mis objetivos ni habría sido posible realizar este trabajo.

A Mónica, la persona que me ha ayudado y apoyado cuando más lo he necesitado en cada paso que he dado, sin ella todo habría sido más difícil. Gracias por ayudarme en esta etapa tan importante de mi vida.

A José Antonio Díaz y Jesús Arias por ayudarme a finalizar este proyecto, gracias a ellos y a su consejo se ha podido desarrollar este trabajo final de grado.

"En la vida hay algo peor que el fracaso:

no haber intentado nada."

Franklin D. Roosevelt

Abstract

There are a large number of different enterprise applications; each application specializes in a different management task. For example, accounting software records and processes accounting transactions within functional modules such as accounts payable, accounts receivable, payroll, and trial balance. Business intelligence (BI) allows a company to gather, store, access, and analyze corporate data to aid in decision-making. Customer relationship management (CRM) allows interaction with current and future customers. Enterprise resource planning (ERP) allows enterprise asset management to facilitate decision-making and to optimize human and material resources with lower risk as well.

Open source software provides an easy way to access to this kind of software for small and medium-sized enterprises. Content management systems (CMS) and business software like Wordpress or Magento have made growth possible for many companies.

The software business provides automation, which allows a company to be managed by only one person. Formerly, lead management medium-size enterprise needed a large staff.

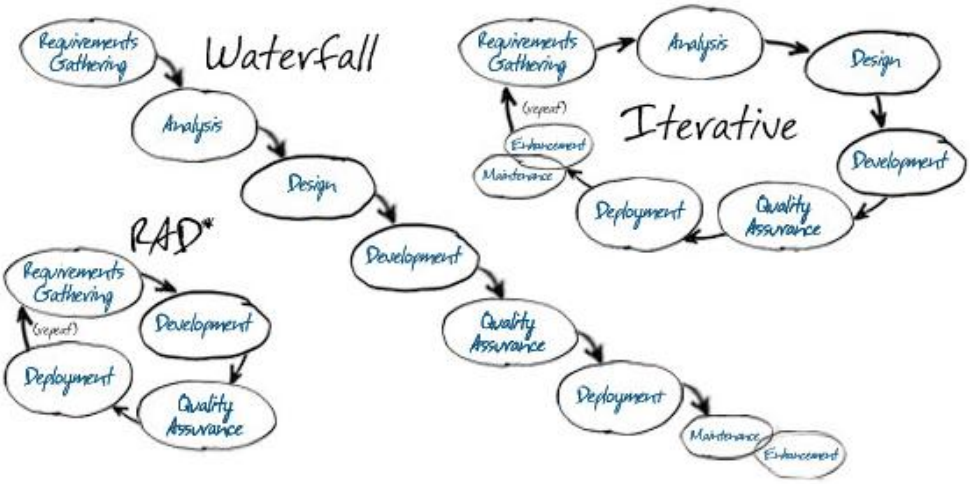
The benefits of having data centralized in different applications are that it minimizes inconsistent data and also requires fewer human resources. Data integration is a necessary factor for any company if it wishes to survive.

The main goal for this final degree is to implement an SME technology platform. For the development of this project, as well as other similar projects, I'm going to design a methodology to carry out projects, which requires that an eCommerce platform and business management software are integrated. This methodology is very important for the final result, if the steps are not explained with detail, developing errors or delays could appear in project time.

The core of this work will be the study of integration technologies like Web Services. Web services are tools that provide a multitude of software interoperability. They are a way to communicate applications or any electronic devices over the internet. It is software that uses standard protocols and can be used in any programming language. This has positioned it as the referent in communication between different

applications thanks to the support of big companies such as IBM or Microsoft.

Using a methodology to develop a project of these characteristics; serves to structure, plan and control the process of software development solutions.



Developing a good software depends on a large number of activities and stages, where the impact of choosing the methodology for a team on a particular project, is critical to the success of the final product.

All methodologies must be adapted to the context of the project (technical and human resources, development time, system type, etc.) Historically, traditional methodologies have tried to address as many situations as possible of the projects, requiring a considerable effort to be adapted, especially in small projects with changing requirements.

For the current project, it will be necessary to study the integration of enterprise applications (EAI). EAI is the use of software to integrate a set of enterprise applications. There are two major topologies: hub-and-spoke and bus.

State of the art

The next step is to create a study of the project's technologies; in this study I begin an investigation of the technologies of communication between software via internet.

First, I investigate Web Services and how they work. Then I emphasize SOAP (simple object access protocol), describing the different parts that compose a SOAP message.

Study of technologies

For this Project, an eCommerce platform and an ERP solution were necessary that allowed expansion of their capabilities.

There is substantial software for development of an online shop like Open Source, private and recurring payment solutions.

I have done a study to determine which eCommerce platforms were the most used. The result was that Magento, Prestashop and Open Cart were the most used.

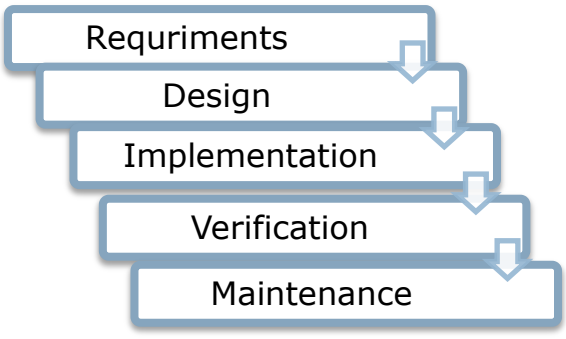
Then a comparison between the three platforms of eCommerce was performed. Magento was the best software because:

- It is Open Source, making it possible to create new extensions to integrate with other software.
- Magento has a bigger community, which is a disposal of the users in forums, wiki, blogs, webinars, etc.
- It supports high web traffic.
  Magento is recognized by all the developers.

After choosing Magento, I started the search of an Open Source ERP, which could integrate with Magento. In this search I found the next ERPs: Openbravo, Dolibarr and OpenERP (also OpenERP is called Odoo).

Methodology

The paradigm chosen for this project is the waterfall model. It is widely used in software development processes and is the paradigm most frequently used today.



The beginning of each stage must wait for the end of the previous stage. The waterfall model Promotes effective working methodology: Define before design, design before implementing.

For the development of the project, the methodology specified above (waterfall) is followed. It was performed using the case of a fictitious company but with real problems.

Certain requirements are created to detail the initial requirements, and then the system design is performed. Next, enterprise applications are installed and finally the integration is developed.

I talk about the goals achieved in the development of the project. Also I detail the different skills that I have been able to reach during these months.

Finally, I detail potential actions that could be made to improve the project. These actions could have developed if the project was more extended in time.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1. Introduction

1.1. Motivation

After the growing demand for tools to develop online stores and facing the difficult situation of SMEs, it has been decided to perform a project to improve processes of midsize companies.

A powerful ecommerce platform will improve the efficiency in time and cost to sell through the internet. An e-commerce platform has many features prepared for its use after the installation so there is no need to invest so much if the purpose is to do something simple using Open Source tools. The difficulty increases when we want to perform an extra functionality in the system.

This project will study the Web technologies that perform system integrations, principally Web Services. The aim of using these elements is to integrate the technology infrastructure management of a SME.

1.2. Goals

For the success of this project, the main goals are the following:

- The first objective is to get data integration between an Open Source ERP and eCommerce platform and then adapt it to the requirements of a SME. This objective aims to improve business processes by providing cost savings and business process automation.
- The second objective is to develop a methodology that can be used for future projects and which details the way by which the project is developed. This will be done by detailing the various stages that the project must pass through.

The secondary goals of this project appear from the main objectives:

- The next goal is to search for enterprise applications that can satisfy the needs of a SME, which can then be adapted or expanded to create communication with other business tools.
- Another goal is to design all the architecture of the suite of a SME. This design must describe how the applications can be integrated.
- The next goal is develop a module that extends the basic functions of the platform of eCommerce, as well as two modules within ERP that can provide synchronization and data integration.
- The ultimate goal is to properly document all stages of the project so that you can serve as reference, where the proposed methodology and its validation are shown in detail.

1.3. Introduction to the document

First, the work plan reflects the life cycle of the current project; the deadlines are represented in a Gantt chart. Additionally, the tasks that compose the project are detailed within WBS scheme.

The state of the art conducts a study of technologies to be used, different software on the market (that meet the needs), plus all applications and competence are discussed, providing a rationale on why they have chosen.

In addition, a methodology has been developed for similar future projects; these projects will require the implementation of technology structure of an SME with an online store and several physical stores.

For the validation of methodology, I'm going to follow a few steps to get a working solution for the implementation. In the validation, first the requirements of the application are detailed, then the design is detailed with diagrams and schemas.

In order to verify the project has reached the objectives, Unit Tests are performed to verify automatically that the code is correct. It has been also necessary to manually check all the use cases of the system.

Then the state of eCommerce and the trends of the actual commerce are described. Also in this section, I describe business software like ERP, which is business management software that companies use to collect, store, supervise, and interpret data from many business activities.

Capítulo 2. Plan de trabajo

A continuación se procede al diseño de un plan de trabajo, que marque las tareas y plazos a seguir en proyecto actual. La planificación del proyecto es una parte fundamental para que este acabe en el plazo correcto, ya que de una buena planificación dependerá un correcto desarrollo. El proyecto está compuesto de cinco fases que se detallan a continuación:

Fases del proyecto			
Nombre	**Descripción**	**Operaciones**	**Duración**
1-Estudio del Problema	En esta fase se realiza el estado del arte y se analizan las tecnologías empleadas en la integración de sistemas.	1. Estudio de necesidades. 2. Estudio EAI. Buscar ERP. 3. Búsqueda eCommerce. 4. Elección: eCommerce y ERP.	4 semanas
2-Planificación	En la siguiente fase se planifica y diseña el proyecto, además se detalla la metodología que se emplea y recomienda para la integración de sistemas.	1. Metodología. 2. Diseño arquitectura general.	3 semanas
3-Ejecución	En la siguiente fase se lleva a cabo el desarrollo, implantación y la integración de un ERP junto a una plataforma de eCommerce aplicando una metodología.	1. Configuración servidor. 2. Desarrollo módulo ERP. 3. Sincronización periódica (ERP). 4. Desarrollo módulo eCommerce.	6 semanas

4-Pruebas	Se lleva a cabo un plan de pruebas para verificar el funcionamiento de la integración.	1. Pruebas básicas. 2. Pruebas finales. 3. Corrección errores.	2 semanas
5-Documentación	Se crea la memoria del proyecto donde se detalla lo estudiado en la fase 1, y se documenta lo desarrollado en la fase 3 y 4.	1. Estado del arte. 2. Diseño. 3. Ejecución.	7 semanas

Tabla 1: Desglose en fases del proyecto

Los paquetes de trabajo a seguir se muestran en el siguiente diagrama WBS (*Work Breakdown Structure*):

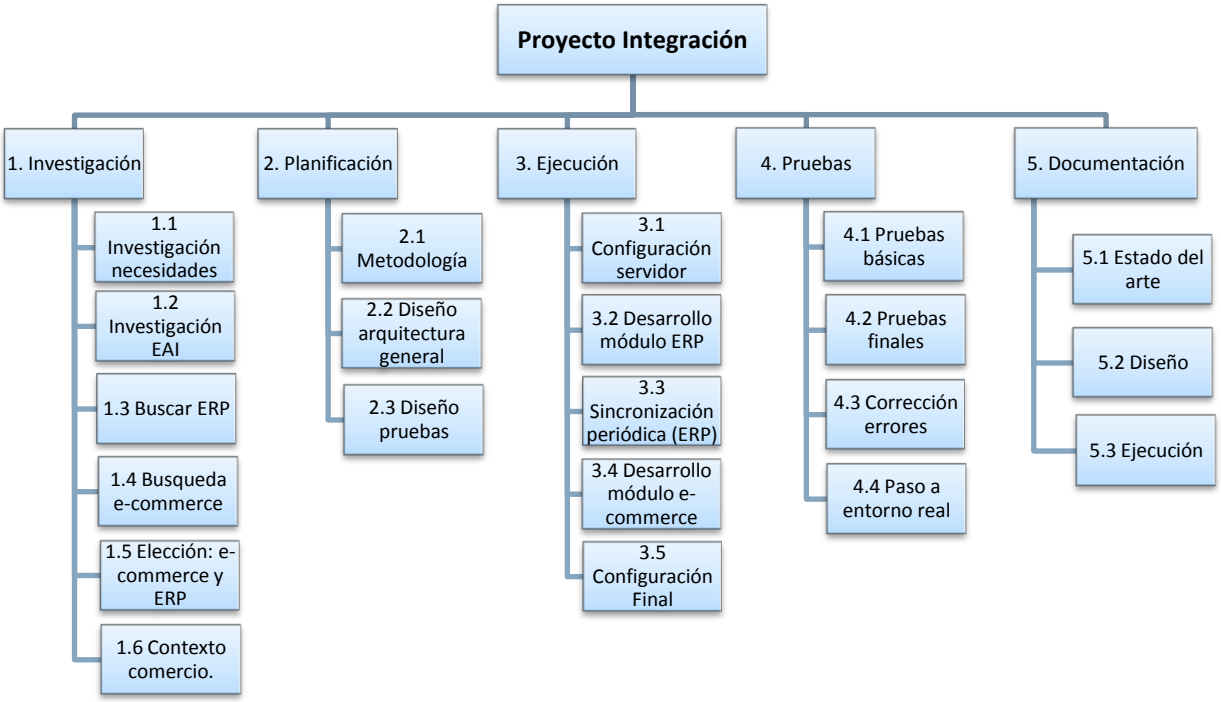


Ilustración 1: Estructura de descomposición de trabajo [WBS]

A continuación se ilustra gráficamente la planificación de las etapas del proyecto. Con el diagrama de Gantt se consigue ver cómo las etapas transcurren a lo largo del tiempo, además de poder ver los distintos hitos a los que se ha de llegar.

Nombre	Fecha de in...	Fecha de fin
proyecto	1/04/14	11/09/14
investigación	1/04/14	30/04/14
investigación necesidades	1/04/14	4/04/14
investigación EAI	7/04/14	11/04/14
búsqueda ERP	7/04/14	16/04/14
búsqueda e-commerce	17/04/14	29/04/14
elección: e-commerce y ERP	30/04/14	30/04/14
fin documentación investigación	1/05/14	1/05/14
planificación	1/05/14	20/05/14
metodología	1/05/14	9/05/14
diseño arquitectura general	12/05/14	20/05/14
fin documentación planificación	21/05/14	21/05/14
ejecución	21/05/14	2/07/14
configuración servidor	21/05/14	23/05/14
desarrollo modulo ERP	26/05/14	17/06/14
sincronizaóion periódica(ERP)	26/05/14	10/06/14
desarrollo modulo e-commerce	18/06/14	2/07/14
desarrollo completado	3/07/14	3/07/14
pruebas	3/07/14	21/07/14
pruebas básicas	3/07/14	9/07/14
pruebas finales	10/07/14	11/07/14
corrección errores	14/07/14	21/07/14
final fase validación	22/07/14	22/07/14
documentación	22/07/14	11/09/14
final proyecto	12/09/14	12/09/14

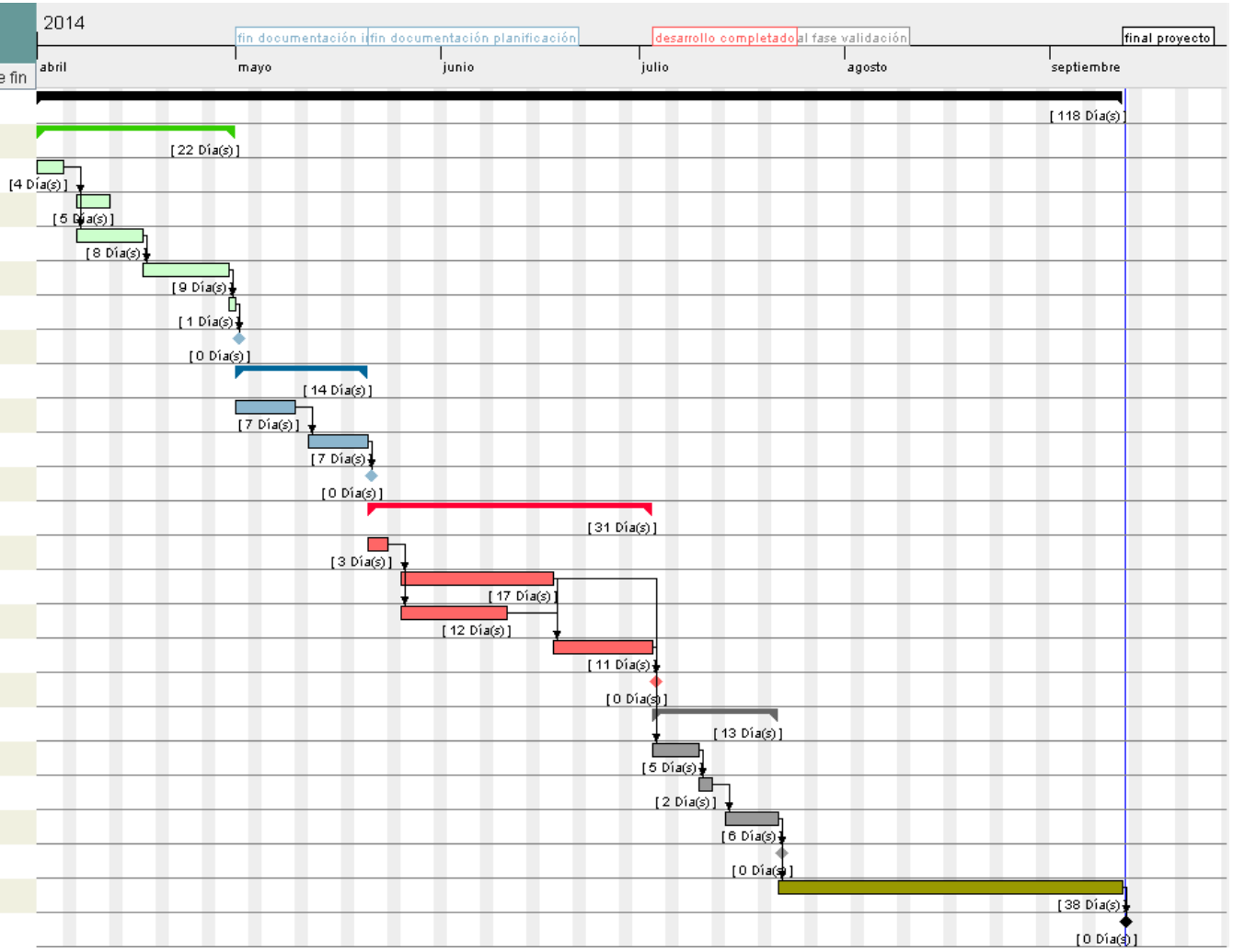


Ilustración 2: Diagrama de Gantt

Capítulo 3. Estado del Arte.

En esta sección del documento, se realiza un estudio sobre las tecnologías relacionadas con la integración entre aplicaciones empresariales. Este capítulo pretende proporcionar unos conceptos técnicos para la mejor comprensión del proyecto.

3.1. Integración de aplicaciones empresariales

Se comienza hablando sobre la integración de aplicaciones empresariales o EAI (del inglés *Enterprise Application Integration*). Este término hace referencia al conjunto de soluciones *software* que se desarrollan para tener conectadas entre sí todas las herramientas *software* de una empresa [1].

Hay una gran cantidad de *software* que puede ser conectado dentro de una empresa, como pueden ser: programas ERP, programas CRM, plataformas de comercio electrónico, herramientas de *business intelligence*, programas para la gestión de recursos humanos y programas de gestión financiera.

Cada uno de estos programas puede necesitar información de otro, pero esta información es privada dentro de cada aplicación. Para otorgar interoperabilidad entre aplicaciones, es necesario implementar un servicio, que cuando reciba una petición de otra aplicación, pueda devolverle la información (se requiere estar autorizado).

La información que almacenan distintas aplicaciones es muy dispar, por lo que será necesario el uso de estándares para implementar una integración. Las implementaciones pueden ser llevadas a cabo usando distintas topologías [2]. Siendo las dos topologías básicas: topología *bus* y topología *Hub & Spoke*.

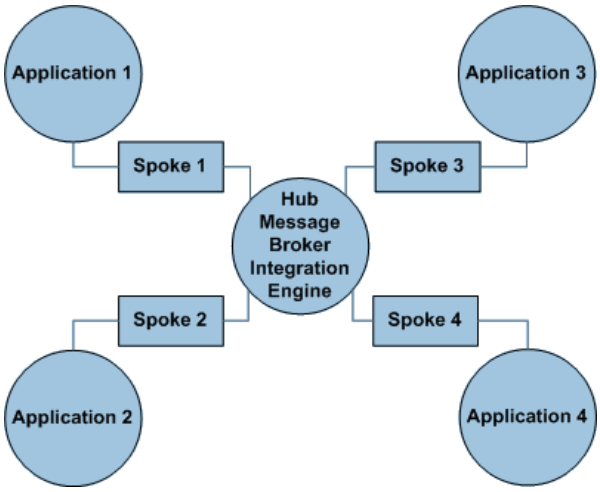


Ilustración 3: Diagrama topología Hub & Spoke [2]

- La topología *Hub & Spoke* usa un nodo centralizado (*hub*) y un número N de adaptadores (*spokes*), los cuales conectan las N aplicaciones al nodo central. Cada adaptador simplemente procesa la información desde la aplicación emisora y la envía hacia el nodo central en un formato compatible con este. El problema puede venir en el momento en el que el nodo central no tenga un *hardware* demasiado potente para procesar los mensajes de las N aplicaciones concurrentemente y falle, dejando todo el sistema paralizado.

- La topología en forma de *bus* hace uso de un canal común a todas las aplicaciones. En el canal, la información se propaga a todas las aplicaciones y solo las que se suscriben a un evento reciben la información. Cuando una aplicación quiere mandar información, esta debe enviársela primero a un filtro (*adapter*) que la convierte a un formato común dentro del bus. Posteriormente este mensaje se vuelve a traducir en un adaptador para llegar a su aplicación destino.

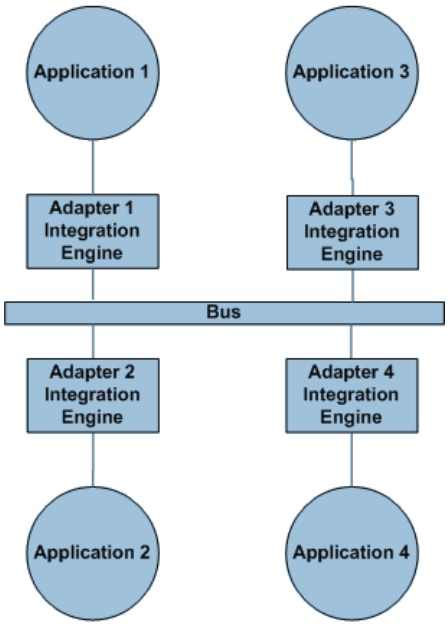


Ilustración 4: Diagrama topología Bus [2]

La principal diferencia que existe entre las dos topologías anteriores, es que la topología *bus* es más escalable que la topología *Hub & Spoke*, debido a que solo es necesario que el mensaje pase por un adaptador para que la información sea compatible para las demás aplicaciones. No obstante, esto hace que la implementación de este tipo de topologías sea más costosa. Este sobrecoste es debido a factores como: la coordinación entre todas las aplicaciones para utilizar un

estándar de mensaje o la creación de un canal de comunicación explícito para el *bus*, lo que hace que esta topología sea implantada en soluciones empresariales propiedad de las grandes empresas de desarrollo *software*, como es el caso de Oracle con su solución: *Oracle SOA Suite*[1], IBM con su solución*: IBM Integration Bus Advanced*[2] o soluciones de código abierto desarrolladas por varias empresas, que conectan aplicaciones de distintos fabricantes como es: *Mule ESB*[3].

Por otro lado la implementación de una topología *Hub & Spoke* es óptima para conectar 2-3 aplicaciones, lo que implica otro tipo de problemática diferente a la topología *bus*.

3.1.1. Problemática EAI

Desde que la integración de aplicaciones empresariales se popularizara sobre los años 2000, las grandes empresas de *software* han tenido dificultades para llevar a cabo proyectos de integración. De hecho, según la consultora Gartner, hasta hace 10 años el 70% de los proyectos sobre integraciones terminaban siendo un fracaso [3]. Pero es actualmente cuando las PYMES pueden permitirse también tener integrados sus distintos sistemas para ahorrar costes y ser más eficientes.

Cuando se hace uso de distintas aplicaciones y comienza una integración, típicamente surgen algunos problemas:

- Por un lado, cada *software* puede estar escrito en un lenguaje de programación distinto, por lo que la forma en la que se comunican es distinta. Una aplicación puede tener incorporado un Servicio Web para la comunicación con el exterior, pero otra puede no tener incorporado este servicio. Además puede darse el caso en el que un desarrollador se enfrente a una aplicación antigua o mal documentada, que le haga perder el tiempo a la hora de una integración.

- Por otro lado, cada herramienta tiene una base de datos muy distinta, en la que cada conjunto de datos no es fácil de mapear entre cada tabla, por lo que uno de los retos de los ingenieros de integración es hallar las semejanzas entre los campos de distintas aplicaciones.

- Por último se tienen los problemas de rendimiento. Hay tipos de comunicación entre aplicaciones que envían mensajes escritos en XML (*Extensible Markup Language*), lo que hace que se envíe mucha información superflua por la red, provocando retardos.

---

[1] *Oracle SOA Suite: http://www.oracle.com/technetwork/es/middleware/soasuite/documentation/oracle-soa-suite-427128-esa.pdf*

[2] *IBM Integration Bus Advanced: http://www-03.ibm.com/software/products/es/integration-bus-advanced*

[3] *Mule ESB: http://www.mulesoft.com/platform/soa/mule-esb-open-source-esb*

3.2. ERP

ERP son las siglas de Enterprise Resource Planning. ERP puede ser descrito como un *software* de gestión de procesos para empresas. Un ERP permite controlar de forma centralizada los diversos procesos de negocios, finanzas, inventario y recursos humanos. Por lo que se mejora la eficiencia de actividades como la producción o el tiempo empleado para procesos. Por ejemplo, una empresa que tiene un control de productos en almacén, materias primas, trabajadores y proveedores. Gracias al ERP será posible gestionar todos estos recursos de forma fácil y centralizada. También el ERP nos ayuda a verificar si todos los productos están disponibles para el correcto desarrollo de la actividad final (venta, transporte…) y en el caso que alguno no esté disponible, se podría anticipar al evento y tomar las medidas adecuadas para el correcto desarrollo de la tarea.

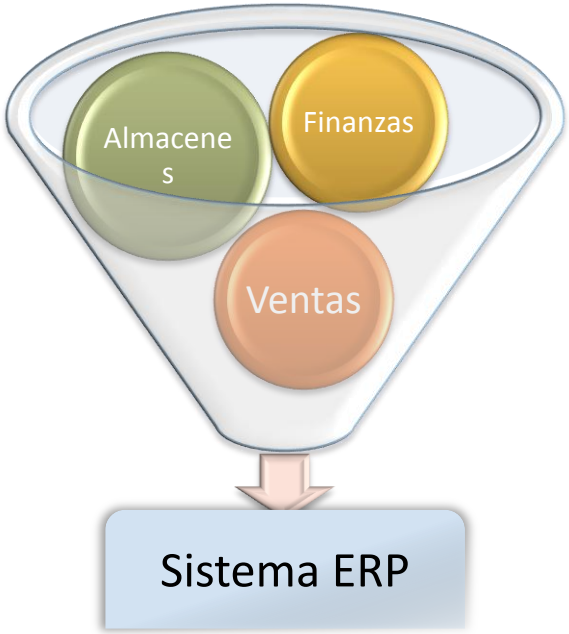


Ilustración 5: Principales elementos que componen un ERP

La incorporación de un ERP a una empresa puede dar unos determinados beneficios, principalmente:

- Disminuir el coste de operación: desde el control del producto y realización de la venta hasta la entrada en efectivo.
- Aumento de la eficiencia, perfeccionamiento de la producción y agilidad en el mercado.
- Aumento de información que tiene la empresa para poner a disposición tanto del personal interno como clientes y proveedores.
- Incremento de la asertividad, ya que se evita la improvisación debido a que ahora se cuenta con información exacta.
- Control total de las distintas tiendas, almacenes e incluso compañías desde un único programa.

El término ERP apareció a principio de los años 90[4], pero hay que remontarse a los 70, años en los que comenzó a utilizarse el software empresarial llamado MRP (material *requirement planning*), el cual había sido usado anteriormente por el ejército americano para gestionar el arsenal y las nóminas. Actualmente en el mercado existen tres tipos de ERP:

- **De código no abierto**: Son sistemas para los cuales se suele requerir el pago de una licencia para su utilización, siendo esta por lo general bastante elevada. El acceso a su código fuente no suele estar disponible para el cliente. Este tipo de sistemas está pensado para empresas grandes que se puedan permitir una inversión de miles de euros. Los mayores desarrolladores de este tipo de *software* son grandes empresas como *SAP*, *Microsoft*, *Sage* u *Oracle*. Estos han desarrollado una solución bastante sólida, madura y con un buen servicio de mantenimiento, lo que les ha permitido dominar el mercado. Por otro lado existen pequeñas empresas que han desarrollado su *software* de código no abierto, que sin haber llegado a conseguir la fiabilidad y la funcionalidad de los grandes sistemas, pueden ser útiles en algún nicho.

  Algunos ejemplos importantes de ERP de código no abierto son:

  - *SAP Business One*[4].
  - *Microsoft Dynamics NAV* (medianas empresas)[5].
  - *Microsoft Dynamics AX* (grandes empresas).
  - *Infor LN*[6].
  - *Sage Murano ERP* (medianas empresas)[7].
  - *Sage ERP X3* (grandes empresas)[8].

- **De código abierto:** Estos sistemas se caracterizan por proporcionar una libertad al usuario final, aportando un beneficio a la empresa que implanta un ERP, ya que no tiene que pagar una licencia sino que sólo paga por la adaptación a sus necesidades. Lo normal es que una empresa que quiere implantar un ERP de código abierto contrate el desarrollo a una agencia especializada, con la ventaja de que en el caso de que esta no fuera de su agrado, podría cambiar por otra empresa que esté especializada en dicho *software*. Proporcionando pues:

  - Libertad en el uso que se da del *software* para cualquier función.
  - Libertad en la edición del código, mejorando la escalabilidad.
  - Libertad para la libre distribución del programa.

---

[4] *SAP Business One: http://www.sap.com/spain/solution/sme/software/erp/small-business-management/overview/index.html*

[5] *Microsoft Dynamics NAV: http://www.microsoft.com/es-es/dynamics/default.aspx*

[6] *Infor LN : http://es.infor.com/product_summary/erp/ln/*

[7] *Sage Murano ERP: http://www.sage.es/software/erp/mediana-empresa/sage-murano-erp*

[8] *Sage ERP X3: http://www.sage.es/software/erp/empresa-internacional/sage-erp-x3*

Algunos ejemplos importantes de ERP de código abierto son:

- *OpenERP* (también llamado Odoo)[9].
- *Openbravo*[10].
- *Dolibarr*[11].
- *Apache OFBiz*[12].
- *Compiere*[13].

Por lo general, todas las empresas pequeñas (entre 10 y 50 empleados) y medianas (entre 50 y 250 empleados), tienen este tipo de aplicaciones, ya que la gestión de todos los elementos de una empresa es mucho más eficiente si se centraliza en un *software* de gestión empresarial.

3.3. Servicios Web

Un Servicio Web (Web Service en inglés), es un estándar de comunicación entre procesos y o componentes [5]. Está diseñado para poder funcionar en cualquier plataforma y lenguaje de programación. Un Servicio Web puede estar programado en Java, PHP (*PHP Hypertext Pre-processor*), C#... y además funcionar en máquinas con *Windows*, *UNIX* o *Linux*. Esta tecnología utiliza un conjunto de estándares: XML, SOAP, WSDL [14] (Web Services Description Language), UDDI (Universal Description, Discovery and Integration) o WS-Security [6], lo que hace posible la interoperabilidad de los Servicios Web.

Existen otros sistemas de comunicación entre aplicaciones distintos de los WS como: CORBA [7] (Common Object Request Broker Architecture), Java RMI (Remote Method Invocation) [8] o WCF basado en .NET *framework* [9], pero generalmente cuentan con más limitaciones que los Servicios Web. La principal ventaja de los Servicios Web sería pues, que están basados en estándares abiertos ampliamente aceptados en la industria, con una gran variedad de herramientas para su uso en los principales lenguajes de programación.

Ya se ha descrito lo que es un Servicio Web, pero ¿para qué sirve? Básicamente permite invocar funciones de otras aplicaciones, sin importar si esa aplicación está en otro ordenador, en cualquier red. No se sabe ni donde esta ese ordenador, ni en qué lenguaje está desarrollada la función que se necesita, pero se sabe que devolverá la información solicitada en el servicio.

[9] *OpenERP:* http://openerpspain.com/
[10] *Openbravo:* http://www.openbravo.com/es
[11] *Dolibarr:* http://www.dolibarr.es/
[12] *Apache OFBiz :* http://ofbiz.apache.org/
[13] *COMPIERE ERP SOFTWARE:* http://www.compiere.com/

[14] Consultar: Anexo 3: WSDL

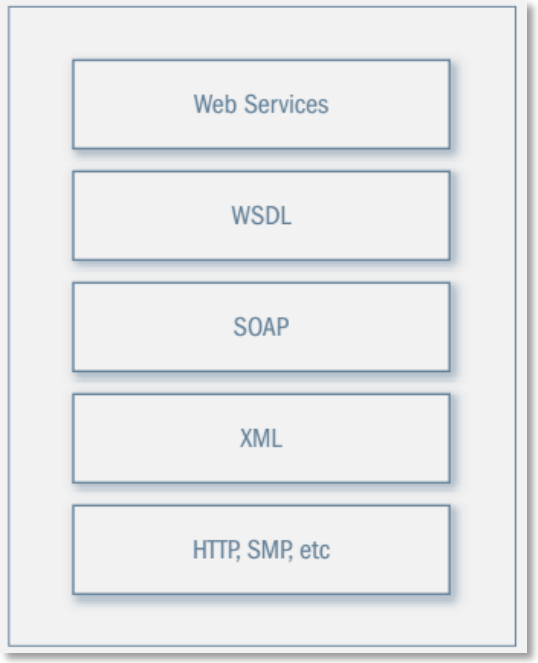


Ilustración 6: Jerarquía de tecnologías y protocolos de los Servicios Web

Uno de los usos más comunes actualmente de los Servicios Web es la implantación en plataformas de comercio electrónico. Estos se utilizan para acciones como verificar los pagos de una tarjeta de crédito con el servidor del banco. El banco de esta manera recibe los datos de la tarjeta y los gastos que se quieren cargar en ella mediante Servicios Web. Dentro del servidor del banco se ejecutan una serie de procedimientos para descontar del saldo, verificar que hay dinero en la cuenta, etc. respondiendo el banco el Servicio Web con información para verificar que todo fue correcto.

Otro uso dentro del comercio electrónico podría ser la recolección de datos entre las distintas aplicaciones de una empresa. Se podría requerir en una aplicación, información de otras aplicaciones de la empresa. Si estas están interconectadas mediante Servicios Web, la información podrá moverse de una aplicación a otra automáticamente.

3.3.1. SOAP

Simple Object Access Protocol [10], es el protocolo en el que se asientan los Servicios Web. SOAP está basado en el estándar XML, lo que no le vincula a ninguna plataforma o lenguaje de programación. Se tiene pues, que SOAP es un protocolo que especifica el formato estándar de la información, para que la comunicación entre aplicaciones sea posible.

SOAP deriva del protocolo XML-RPC (remote procedure call) creado en 1998 por el desarrollador Dave Winer. La creación de SOAP (mayo, 2000) fue en un principio llevada a cabo a manos de *Microsoft*, *Develop Mentor Inc*, *Userland Software Inc* e *IBM* entre otros, estando ahora a cargo del World Wide Web

Consortium (W3C). SOAP es actualmente el método más extendido para la comunicación entre aplicaciones.

A continuación se detalla la estructura de un mensaje SOAP:

- ***SOAP Envelope Element:*** Es el elemento básico de un mensaje SOAP. Este elemento identifica los documentos XML como un mensaje SOAP.

- ***SOAP Header Element:*** Es un elemento opcional, pero es muy útil para mandar meta información acerca de los mensajes SOAP. Este contiene información específica de la aplicación, como la autenticación, paso de directivas o de información contextual relativa al procesado del mensaje. El *SOAP header element* ha de ser el primer hijo del *Envelope element*.

- ***SOAP Body Element:*** Este elemento obligatorio contiene el mensaje que será transmitido entre dos aplicaciones. El *Body element* contiene descripciones sobre el tipo de petición realizada por el cliente.

  El *Body element* también puede contener un elemento opcional: el *Fault element,* el cual contiene información de errores SOAP.

En el siguiente código [11], se muestra el esqueleto completo de un mensaje SOAP.

```
<?xmlversion='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

      <env:Header>
            <m:reserva xmlns:m="http://empresaviajes.example.org/reserva"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
            env:mustUnderstand="true">


             <m:referencia>
             uuid:093a2da1-q345-739r- ba5d-pqff98fe8j7d
             </m:referencia>
            <m:fechaYHora>2001-11-29T13:35:00.000-05:00</m:fechaYHora>
            </m:reserva>
            <n:pasajero xmlns:n="http://miempresa.example.com/empleados"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
            env:mustUnderstand="true">
            <n:nombre>Juan Sanchez</n:nombre>
            </n:pasajero>
      </env:Header>

      <env:Body>
      <p:aclaracionItinerario
      xmlns:p="http://http://empresaviajes.example.org/reserva/viaje">

            <p:ida>
                  <p:salida>
                        <p:eleccionAeropuerto>
                                    JFK LGA EWR
                        </p:eleccionAeropuerto>
```

```
                    </p:salida>
              </p:ida>

              <p:vuelta>
                    <p:llegada>
                    <p:eleccionAeropuerto>
                                  JFK LGA EWR
                    </p:eleccionAeropuerto>
                    </p:llegada>
              </p:vuelta>

      </p:aclaracionItinerario>
      </env:Body>
</env:Envelope>
```

Ilustración 7: Ejemplo código XML/SOAP

La ilustración inferior refleja en forma de bloques la estructura de un mensaje SOAP, esta estructura representa el código del Anexo 2.

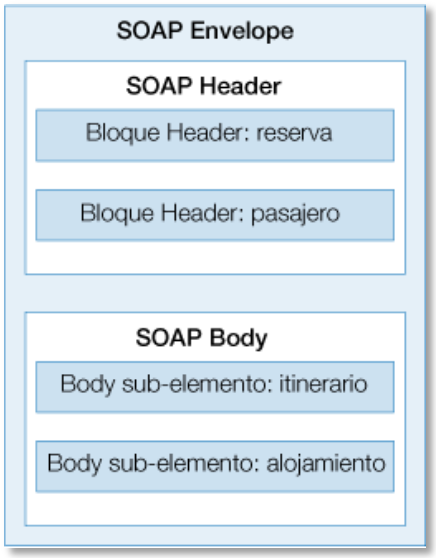


Ilustración 8: Ejemplo mensaje SOAP [11]

A continuación se enumeran las ventajas y desventajas de la utilización de SOAP.

Ventajas de SOAP:

- Neutralidad de lenguaje. SOAP puede ser implementado en cualquier lenguaje o plataforma.
- Uso de XML, logrando una gran adaptabilidad a cualquier entorno y una gran solidez.
- Simplicidad de mensajes, al utilizar XML los mensajes contienen un formato simple.
- Gran facilidad para atravesar cortafuegos y proxies al extender el protocolo HTTP (Hypertext Transfer Protocol).

Desventajas de SOAP:

- Velocidad más lenta con respecto a otros protocolos de comunicación, debido al uso de XML.
- Aumento del consumo de ancho de banda, debido al formato de XML.
- Los datos binarios se codifican a texto, si estos son grandes el proceso puede ralentizarse.
- Dificultad para entender las especificaciones del protocolo.

A continuación se muestra un ejemplo práctico del uso de los Servicios Web en conjunto con SOAP. En la ilustración se muestran los mensajes que se producen en la compra de un viaje en la web de una agencia de viajes.

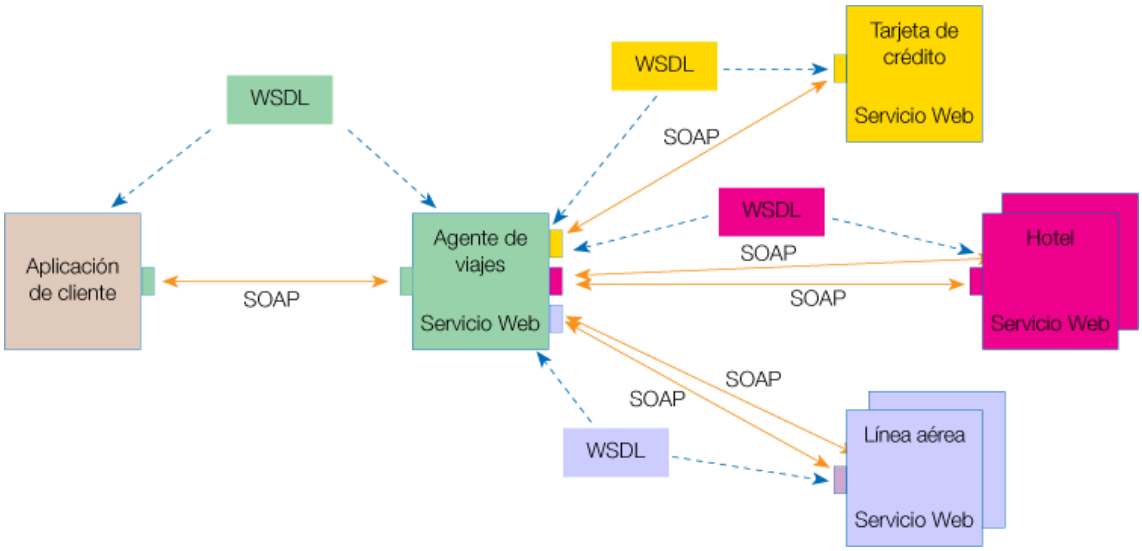


Ilustración 9: Ejemplo de uso SOAP y WS [11]

El proceso de la ilustración superior muestra como la "Aplicación de cliente", se correspondería con el navegador de la persona que está buscando un viaje. Este cliente al consultar información sobre cierto paquete que incluye: Hotel + Avión, está consumiendo en realidad un Servicio Web del servidor de la agencia de viajes. La información que se envía cuando se consume este Servicio Web, se manda con formato de mensaje SOAP. Pero el servidor de la agencia de viajes es un intermediario, por lo que de forma paralela, el servidor de la agencia de viajes está actuando como cliente de Servicios Web, consultando la información del vuelo y del hotel a los Servicios Web de los servidores de la compañía aérea y la empresa hotelera respectivamente. Obteniendo (en el servidor de la agencia de viajes) de vuelta la información del Hotel. No obstante los mensajes SOAP que se han intercambiado, no serían posibles sin que entre ambas partes hubiesen acordado los detalles de los mensajes y su contenido.

Se puede concluir que SOAP es un protocolo de mensajes entre máquinas, el cual especifica el tipo y formato de mensajes que se necesita para acceder a los objetos requeridos. Siendo además, un protocolo que aporta interoperabilidad a aplicaciones creadas en distintos lenguajes de programación. Esto le ha convertido en el método más popular de comunicación entre sistemas a través de Internet.

Capítulo 4. Caso de estudio

A continuación se plantea el caso de estudio del actual proyecto. Este caso surge de la necesidad de una empresa real, pero se va a ocultar cualquier información que la pueda identificar por motivos de confidencialidad. Se hará mención en su lugar una empresa ficticia llamada SOL-España S.A.

Se tiene que la empresa SOL-España S.A. Una mediana empresa de reciente creación, en la cual se producen gafas de sol exclusivas. Quiere vender a través de Internet. Además, esta empresa se encarga de vender directamente al cliente final. Lo hace mediante la tienda *online* que tiene, pero esta tienda está desactualizada tanto funcionalmente como en temas de diseño.

Ilustración 10: Estructura de la empresa SOL-España

El inventario cambia constantemente debido a la existencia de muchos modelos exclusivos de los cuales se fabrican pocas unidades. La marca en este año está muy solicitada por los clientes finales debido a la tendencia existente, lo que provoca cambios rápidos en la disponibilidad de ciertos productos.

SOL-ESPAÑA S.A. tiene actualmente dos tiendas físicas en España. Estas tiendas están vendiendo de media 45 gafas diarias, cada una. Además la empresa

en los próximos meses, tiene prevista la apertura de diez tiendas físicas más. Por otra parte la empresa también va a vender a clientes mayoristas.

Uno de los mayores problemas existentes, es cuando se realizan pedidos en la tienda *online,* de los cuales no hay existencias por ser productos exclusivos. En estos casos el pedido tiene que ser cancelado. Entonces, dado que el problema de la falta de stock está haciendo a la compañía perder ventas, además de perder credibilidad ante los usuarios, se decide desarrollar un *software* que solucione el problema permitiendo tener el inventario perfectamente sincronizado para satisfacer todos los pedidos. Para ello necesitan tener integrada la tienda *online* con los TPV (terminal punto de venta) de las tiendas físicas a través de un ERP y conseguir:

- Alcanzar una posición competitiva en plazos de servicio.
- Disponer de información en tiempo real para la toma de decisiones debido a la implantación de un cuadro de mando logístico.
- Mejorar los costos de los procesos administrativos.
- Mejorar la satisfacción del cliente gracias al mejor servicio prestado.
- Aumento de la satisfacción del equipo humano gracias a tener procesos colaborativos, claramente definidos, comunicados e implantados.

Necesidades

Tras analizar el caso de estudio, se encuentran las necesidades iniciales de buscar un ERP y una plataforma de comercio electrónico. El presupuesto no puede dispararse debido a que el cliente es una PYME. Actualmente tiene dinero para invertir en el desarrollo de la plataforma tecnológica de la empresa, pero no quiere incurrir en gastos innecesarios.

El *software* ERP debe permitir ampliar sus funcionalidades, para así poder conectarse mediante Servicios Web a la plataforma de eCommerce, es decir, se requiere que exista interoperabilidad entre aplicaciones. Además es conveniente que el ERP tenga la posibilidad de incorporar un terminal de venta, para poder instalar en las tiendas físicas. Por otro lado, existe la necesidad de buscar un nuevo gestor de comercio Web. Se pretende actualizar la imagen de la Web que se encuentra actualmente, mediante la instalación de un *software* de comercio electrónico más eficaz. De esta manera se podrán realizar pedidos de forma más ágil. La comparación de tecnologías se detalla en el Capítulo 6: Estudio de tecnologías. En ese capítulo se detallarán y compararán las distintas soluciones del mercado.

Capítulo 5. Propuesta de Metodología

El desarrollo de este proyecto tiene como objetivo detallar una metodología, la cual pueda ser utilizada para proyectos de implantación e integración de software empresarial para PYMES. Es importante detallar con claridad los pasos de esta metodología, ya que de ella dependerá el desarrollo de este proyecto. Además se espera poder ofrecer como referencia el actual documento, para proyectos futuros de características similares. La metodología será probada en un caso de uso real, para poder verificar, si es factible cumplir los requisitos iniciales siguiendo las etapas definidas.

Para proceder a la correcta creación de un proyecto de implantación e integración de herramientas de gestión de empresas de código abierto, es recomendable seguir una serie de pautas para que el proyecto tenga más probabilidad de ser viable y acabar con éxito.

En este trabajo se propone una metodología que PYMES, de entre 10 y 200 empleados, en situaciones similares a la descrita en el caso de estudio del Capítulo 4, puedan aplicar para disponer de una plataforma tecnológica que dé soporte a una tienda web y una o varias tiendas presenciales.

Se plantea que la metodología siga el paradigma del desarrollo en cascada. Se pueden apreciar las distintas fases en la ilustración inferior.

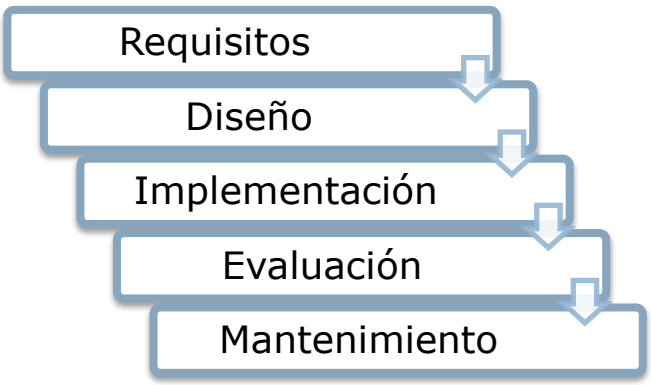


Ilustración 11: Metodología de Cascada

La primera fase de la metodología es establecer una serie de requisitos, en los cuales se especifique la dimensión de la integración. Para la elaboración de los requerimientos es necesario conocer la arquitectura y herramientas de las distintas aplicaciones. Por ello se recomienda realizar una preparación técnica sobre los elementos a integrar. El estado del arte del actual documento, junto a sus referencias, proporcionan los conceptos básicos para conseguir una preparación técnica.

En la elaboración de **requisitos** se han de listar las distintas necesidades del cliente para la aplicación. Para la elaboración de un documento de requisitos se recomienda crear una lista de requisitos funcionales, que especifiquen las acciones que ha de implementar el sistema, y una lista de requisitos no funcionales que establezcan las limitaciones y condiciones del sistema.

Para la especificación del formato de un requisito, este constará de una tabla con los siguientes elementos:

- **Identificador:** indica si es requisito funcional (RF) o no funcional (RNF), seguido de un identificador numérico.

- **Nombre:** nombre breve y claro del requisito.

- **Prioridad:** define el estado de prioridad que tiene el requisito, puede ser:
  - Opcional: si se cumple el requisito, es bueno para el resultado final del proyecto.
  - Obligatorio: el proyecto para estar finalizado debe cumplir este tipo de requisito.

- **Dificultad:** define el nivel de dificultad estimado para que se pueda llevar a cabo este requisito.

Identificador	**RF-XX**		
Nombre	<< nombre requisito >>		
Descripción	<< descripción >>		
Prioridad	Obligatorio/Opcional	**Dificultad**	Bajo/Medio/Alto

Tabla 2: patrón de requisito

Además, es recomendable elaborar un conjunto de diagramas de casos de uso, para que la fase de diseño e implementación sea más detallada. Para la elaboración de estos, se recomienda que estén compuestos de los siguientes elementos:

- **Actor:** un actor es un rol que desempeña un usuario o una entidad en el sistema.

- **Casos de uso:** es la tarea producida en consecuencia de la acción de algún agente externo, ya sea un actor u otro caso de uso.

- **Relaciones:** es el vínculo entre actores y casos de uso o entre casos de uso, se utilizan tres tipos:
    - **Asociación:** La forma más básica de relación, representa la invocación desde un actor o un caso de uso a otro actor u otro caso de uso. Se representa con una flecha simple.
    - **Dependencia o Instanciación:** Es una forma de relación entre casos de uso, en la cual un caso de uso depende de otro, es decir se crea. Se suele representar con una flecha discontinua.
    - **Generalización:** Representa la herencia entre distintos casos de uso.

- **Escenario:** Contexto en el que se desarrolla el caso de uso (ilustración inferior).

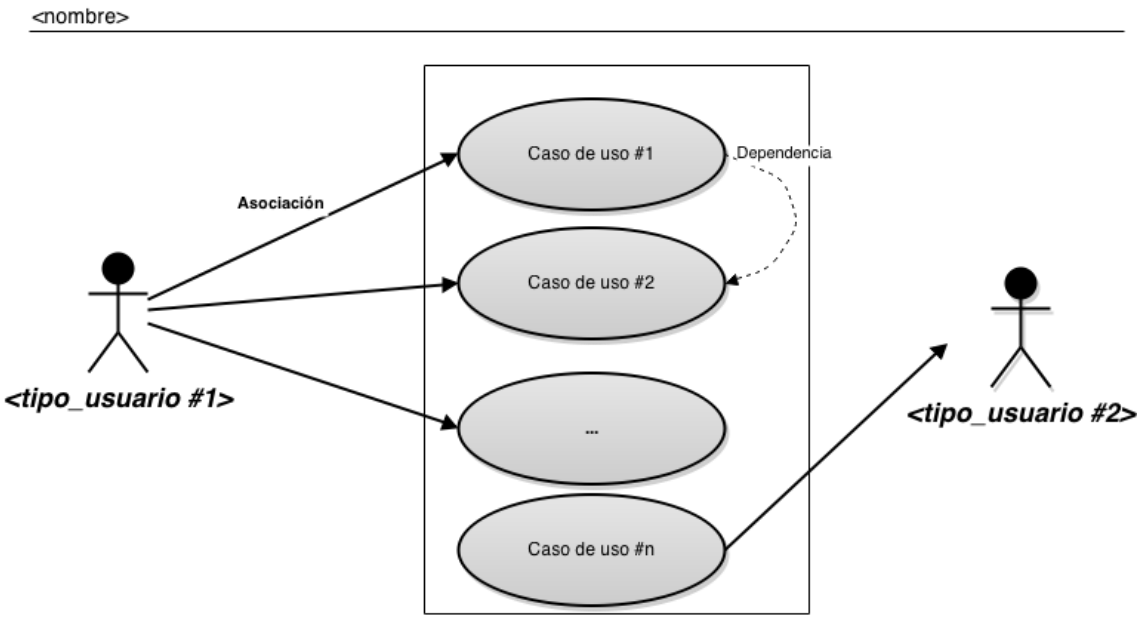


Ilustración 12: Ejemplo nomenclatura caso de uso.

Además para detallar los diagramas se utilizarán tablas con la información:

- **Identificador:** Identifica al caso de uso.
- **Nombre:** Nombre del caso de uso.
- **Actor:** Actor o elemento que provoca la acción.
- **Objetivo:** Objetivo que se consigue realizando el caso de uso.
- **Precondiciones:** Condiciones previas que se han de dar para realizar la acción.
- **Postcondiciones:** Condiciones que se dan una vez realizada la acción.
- **Escenario básico:** Pasos que se llevan a cabo para realizar la acción.

Una vez se tiene el alcance de la implantación e integración. Se procede al **diseño** del sistema.

El siguiente paso sería realizar un diseño de la topología de los distintos sistemas. Para ello, según lo descrito en el apartado 3.1, se puede elegir una topología en forma de *bus* o una topología *Hub & Spoke*. Lo recomendable en una PYME con dos o tres sistemas, es hacer uso de una topología *Hub & Spoke*, donde el ERP sea el nodo central. En base a la topología se debe diseñar un esquema detallado de la arquitectura, similar al de la ilustración inferior.

Ilustración 13: Ejemplo topología con ERP central.

En un sistema de este tipo, se tienen distintas bases de datos en cada aplicación. En cada una habrá un modelo de datos distinto, porque un ERP no necesita la misma información que un comercio *online*. Por ello se establecen los campos que se quieren tener centralizados. Después se realiza un esquema donde aparezcan las estructuras de datos que se van a tener vinculadas en ambos sentidos, ejemplo de esquema de sincronización:

	ERP → eCommerce	**eCommerce → ERP**
Productos	✓ Si	× No
Clientes	× No	✓ Si
…	…	…

Tabla 3: Ejemplo de esquema sincronización

Cuando se tienen seleccionados los datos de la sincronización, es necesario organizar los flujos de información entre aplicaciones, para así poder definir con más criterios las necesidades de la integración.

La integración entre sistemas se realizará mediante Servicios Web. Por ello la principal área de investigación será la implementación de estos servicios y su funcionamiento. Estos pueden ser desarrollados utilizando distintos lenguajes como: PHP, Java, C o .Net entre otros.

Una vez concluida la parte de diseño, se da paso a la **implementación**. En primer lugar es necesario un servidor *Web* donde almacenar las aplicaciones. Este debe ser configurado adecuadamente para que pueda soportar las conexiones de este tipo de solución. Posteriormente se instalan las aplicaciones y se configuran.

En este punto, es necesario seleccionar la forma en la que realizar la integración requerida. Tanto las plataformas de comercio electrónico como los ERP se caracterizan por extenderse mediante módulos, por lo que es necesario seleccionar los módulos que se utilizarán para la integración de datos entre aplicaciones. Los criterios para elegir el módulo correcto son:

- En primer lugar, comprobar que el ERP o eCommerce no disponen de módulo de integración incorporado. Algunos tipos de *software* con poco mercado, se integran con facilidad con las aplicaciones más usadas para atraer usuarios. Si el *software* seleccionado no contiene ningún módulo de integración será necesario buscar módulos externos.

- En segundo lugar, buscar las soluciones de integración existentes (de pago o gratuitas) en cada aplicación. Existen tiendas oficiales de extensiones en las cuales se ofrecen funcionalidades fácilmente instalables que proporcionan un gran rendimiento. Implementar este tipo de solución para lograr una integración plena es caro (más de 10.000 €), por lo que se desarrollan con el objetivo de poder vender muchas con facilidad. Estas extensiones se venden con un coste muy inferior al de su desarrollo. Un ejemplo sería el siguiente módulo de Magento: OpenERP Bridge [12], un módulo que mediante la API (Application Programming Interface) de REST mantiene sincronizados productos, clientes y pedidos. Tiene un coste de 199 $ y se ha vendido más de 200 veces, pero aproximadamente el coste de desarrollo y mantenimiento pueden ser 30 veces el precio de compra. Desarrollar un módulo útil puede generar unos grandes beneficios.

- En tercer lugar, si se requiere un alto grado de personalización en la integración o no existe ninguna extensión que pueda realizar lo requerido, hay que desarrollar uno o varios módulos que cumplan con los requisitos. La implementación depende de diversos factores pero por pequeña que sea, el desarrollo de una integración es complejo, por lo tanto, habrá que realizar un plan de diseño para el desarrollo.

El desarrollo de cada módulo de integración, ha de tener en cuenta que el envío de datos se puede producir periódicamente (una vez al día, cada hora...) o de forma asíncrona. Realizarlo de forma asíncrona es más difícil, ya que diversas tareas pueden modificar cierta información en la BBDD. Por ejemplo el stock de un producto puede ser modificado dentro de un comercio *online* por: una venta, una devolución, una modificación de pedido, un movimiento de inventario o una petición a la API. Mientras que el inventario en un ERP puede ser modificado por: un movimiento de almacén, una venta, una modificación manual, un evento externo...

El desarrollo de los módulos ha de lograr ser compatible con otras implantaciones, es decir, hay elementos genéricos entre aplicaciones que pueden estar sincronizados.

A continuación se muestra el flujo de funcionamiento de los posibles módulos a implementar:

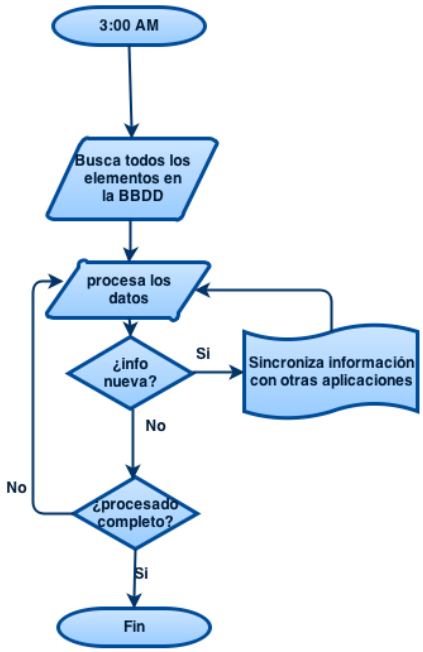


Ilustración 14: Diagrama de flujo sincronización periódica.

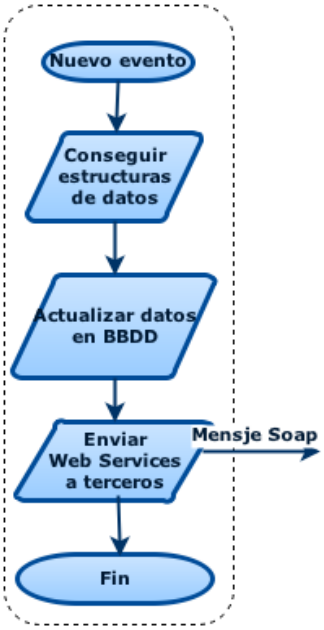


Ilustración 15: Diagrama de Flujo de módulo asíncrono.

Un módulo asíncrono tiene toda la información sincronizada en tiempo real. Esto requiere más recursos disponibles en el servidor para soportar compras concurrentes, además de una implementación más compleja. Por el contrario, un módulo que realiza la sincronización de forma periódica simplemente actualiza toda la información con una frecuencia fija establecida.

Una vez estén instalados los nuevos módulos, sería necesario configurar las direcciones de las demás aplicaciones. Típicamente será suficiente con especificar la ruta del WSDL.

La siguiente fase de la metodología sería el diseño de un plan de pruebas para **evaluar** si se cumplen los objetivos establecidos al inicio de la metodología. Las etapas de esta fase de evaluación deberían ser:

- Diseño de pruebas: elaborar un plan de pruebas que analice el cumplimiento de los requisitos funcionales y no funcionales. Además hay que comprobar desde la vista de la aplicación las correctas fases de la integración.
- Implementación y ejecución de las pruebas: se han de llevar a cabo pruebas unitarias y pruebas funcionales.

- Evaluación de criterios de salidas: se han de evaluar los resultados obtenidos en la etapa de ejecución.

Estas pruebas han de evaluar la mayor cantidad de funcionalidades posible, para poder aumentar la calidad del producto final.

Un aspecto también importante para la finalización del proyecto sería definir con un diseñador gráfico la estilización del portal de comercio electrónico. Actualmente Magento cuenta con multitud de temas o plantillas que cargan un diseño predefinido. Las plantillas son una interfaz que sustituye a la inicial, las cuales utilizando HTML y CSS cambian totalmente la imagen del sitio *Web*. En tal caso se recomienda acudir al sitio w*eb* www.themeforest.net donde se pueden encontrar temas y plantillas de un nivel profesional para Magento y otro tipo de plataformas.

Una vez finalizada la implantación se requiere un trabajo constante de **mantenimiento**. Al igual que en una tienda física se trabaja continuamente para mejorar el aspecto del local. En un portal de comercio electrónico es necesario un trabajo de marketing para que los clientes queden satisfechos y puedan transmitir las ventajas de la tienda. Además es necesario un trabajo en SEO (Search Engine Optimization), lo que ayudaría a posicionar en los buscadores como Google.

En lo que a mantenimiento informático se refiere, se necesita un mantenimiento de servidor, así como un sistema de copias de seguridad recurrente que salve todos los datos de la empresa. Será necesario especificar unas tareas de mantenimiento.

Además, en ocasiones será necesario expandir las funcionalidades del comercio para mejorar la integración con los envíos, poner nuevos métodos de pago, nuevas secciones, etc.

Esta metodología es válida para las empresas medianas, que puedan acaparar un gran volumen de venta con poco personal, y que además quieran tener presencia para vender tanto en tiendas físicas como en su propia plataforma de comercio electrónico.

Capítulo 6. Estudio de tecnologías

En este capítulo, se va a proceder a detallar una serie de procedimientos para guiar la selección de herramientas. Estas herramientas se recomiendan para seguir la metodología que se ha elaborado en el capítulo anterior.

6.1. Búsqueda y selección de ERP

En primer lugar, se procede a realizar un proceso de selección de un *software* ERP. Se realizan la selección entre *software* de código abierto por dos motivos. Por un lado, porque las aplicaciones empresariales de código privado, tienen una licencia costosa, que impide a las PYMES realizar un desembolso inicial tan grande. Y por otro lado, porque la formación para el desarrollo de estas, es menos accesible o muy costosa. Tras buscar los ERP de código abierto en el mercado[15], se vio que la solución óptima sería la implantación de un *software* ERP de código abierto. A continuación se muestra la popularidad en el tiemplo, de los principales ERP del mercado.

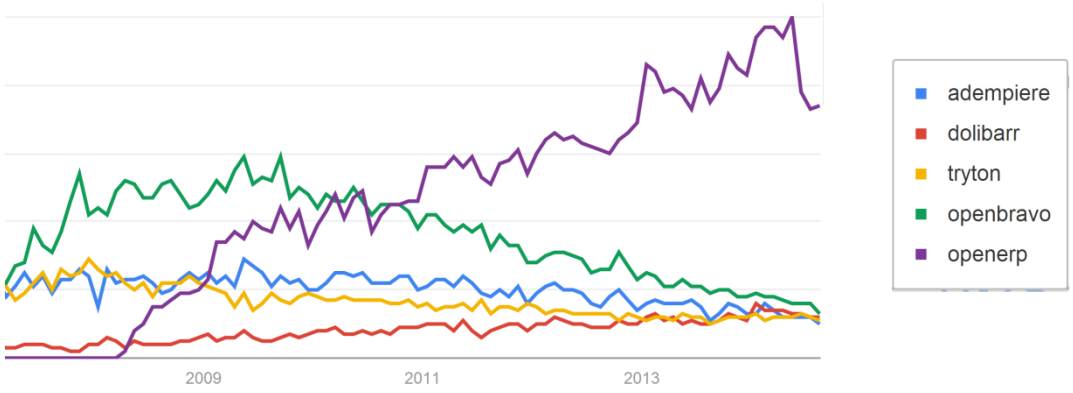


Ilustración 16: ERP de código abierto más populares. GoogleTrends

Tras ver el mercado actual y su tendencia. Se hace una comparación funcional entre los 3 ERP posibles:

[15] Anexo 4: Lista programas ERP de código abierto

		Dolibarr ERP/CRM	OpenERP	openbravo
Detalles Generales	**Requisitos para la instalación**	PHP, MySQL Servidor Web	Python y PostgreSQL	Java, PostgeSQL, Servidor de aplicaciones.
	Modo de implantación	Sí Cloud Computing. Servidor dedicado.	Servidor dedicado	Servidor dedicado
Precio	**Precio inicial (versión community)**	0€	0€	0€
	Precio solución SaaS (Cloud Computing)	9€/mes/usuario-Básica 42€/mes (Avanzada)	12€/mes/usuario-Básica 99€/mes (Avanzada)	39€/mes/usuario
	Licencia	Open Source	Open Source	Open Source
Funcionalidades Básicas	**Funcionalidad contabilidad**	Gestión de cobros: No Gestión de Pagos: No Contabilidad general: SI Multidivisa: Sí	Gestión de cobros: Sí Gestión de Pagos: Sí Contabilidad general: SI Multidivisa: Sí	Gestión de cobros: Sí Gestión de Pagos: Sí Contabilidad general: SI Multidivisa: Sí
	Pagos y facturas	-Transacciones automáticas: No -Procesado de tarjeta de crédito: Sí -Facturas: Sí -Pagos recurrentes: Si -Gestión de compras: Sí	-Transacciones automáticas: Sí -Procesado de tarjeta de crédito: No -Facturas: Sí -Pagos recurrentes: Si -Gestión de compras: Sí	-Transacciones automáticas: Sí -Procesado de tarjeta de crédito: No -Facturas: Sí -Pagos recurrentes: No -Gestión de Compras: Sí
	Reportes y Estadísticas	Balances: Sí Informe de Ventas: Sí Bussines Intelligence: No	Balances: Sí Informe de Ventas: Sí Bussines Intelligence: No	Balances: Sí Informe de Ventas: Sí Bussines Intelligence: Sí

Funcionalidades Adicionales	**Relaciones Bancarias**	-Balance de comprobación: Si -Conciliación Bancaria:No -Sincronización cuentas bancarias: No	-Balance de comprobación: -Conciliación Bancaria:Sí -Sincronización cuentas bancarias: Si	-Balance de comprobación: No -Conciliación Bancaria: Sí -Sincronización cuentas bancarias: No
	Reglas de impuestos	IVA: Sí VAT: Sí	IVA: No VAT: Sí	IVA: Sí VAT: No
	Módulos instalados.	-Bussiness Intelligence: Si -Gestión Inventario: Si -Gestión de Nóminas: Si TPV integrado: Si CRM: Si	-Bussiness Intelligence: No -Gestión Inventario: Sí -Gestión de Nóminas: Si TPV integrado: Si CRM: Si	-Bussiness Intelligence: Si -Gestion Inventario: Si -Gestión de Nóminas: No TPV integrado: No (se puede integrar fácilmente) CRM: Si
	Eventos y Trigger	Sí. Desarrollo mediante módulos.	Si *	Si *
	Web Services	SI. XML-RPC,SOAP	SI XML-RPC	SI, REST
Comunidad y Soporte	**Origen:**	Francia	Estados Unidos/Holanda	España / Estados Unidos (SaaS)
	Soporte	-Wiki: Si -Blog oficial: Si -White Papers: No -Webinars: Si -Actualizaciones: bienales -Formulario de contacto: No -Manual de usuario: Si -Foros: Si	-Wiki: No -Blog oficial: Si -White Papers: No -Webinars: No -Actualizaciones: cada 1-2 años -Formulario de contacto: No -Manual de usuario: No -Foros: No	-Wiki: Si -Blog oficial: Si -White Papers: Si -Webinars: Si -Actualizaciones: pocas. -Formulario de contacto: No -Foros: No
Plataforma	**Plataformas Soportadas**	Mac: Si Online: Si Móvil: Si Windows: Si Linux: Si	Mac No Online No Móvil: No Windows Si Linux Si	Mac Si Online Si Móvil: No Windows Si Linux Si

Tabla 4: Comparativa Dolibarr vs OpenERP vs Openbravo

Tras analizar Dolibarr, OpenBravo y OpenERP en profundidad, se ve que el más completo en número de funcionalidades es OpenBravo. Luego, con menos funcionalidades que este, se encuentran Dolibarr y OpenERP. Aunque estos últimos son muy parecidos funcionalmente. Los tres están diseñados para alojarse en un servidor web. Esto es imprescindible para la gestión de una empresa hoy en día.

OpenERP está desarrollado en Python, OpenBravo en Java y Dolibarr en PHP. Según lo experimentado el desarrollo más lento sería el de Openbravo. Mientras que el más manejable por los desarrolladores es Dolibarr. La comunidad de Dolibarr tiene más presencia en América Latina, además participan en ella tanto empresas como particulares. La comunidad de Openbravo se encuentra en España e India, y es bastante más conservadora en el aprendizaje para el desarrollo que las demás. Por último la comunidad de OpenERP es la más pequeña. Por lo tanto la mejor comunidad es la correspondiente a Dolibarr.

Los tres software son recomendables para una PYME, pero el más cómodo de usar es OpenERP seguido por Dolibarr, Openbravo resulta bastante complejo de utilizar.

Finalmente se decidió usar el ERP Dolibarr. La razón principal por la que se eligió Dolibarr, fue porque era posible el desarrollo de módulos de integración con otros sistemas. Además, Dolibarr tenía incorporado un CRM y un TPV, lo que hacía mucho más completa la plataforma en su conjunto. Otros de los motivos por los que se eligió Dolibarr fueron:

- El **coste de implantación**: la implantación de Dolibarr es muy rápida. Necesita un servidor *Web* apache, una base de datos MySql y un servidor en el que alojarse. Además no tiene ningún coste de licencia.
- **Facilidad para los usuarios**: la navegación de Dolibarr es sencilla para el usuario final, se ha podido comprobar cómo un usuario sin experiencia con el ERP Dolibarr, se ha podido manejar en muy poco tiempo con soltura.
- **Modularidad** ágil: Dolibarr tiene una gran capacidad para incorporar nuevos módulos, lo que está haciendo que la comunidad de desarrolladores crezca muy rápido.
- Se puede implantar en la **nube:** la agilidad que ofrece un ERP en la nube para una empresa que vende o gestiona desde distintos puntos.

A continuación, se enumeran las soluciones ERP descartadas para la realización del proyecto actual. Se detallan además las razones de su **descarte** en orden de decisión:

- **Soluciones ERP sin licencia de código abierto**: Este tipo de solución se descartó debido al coste de implantación. El presupuesto no permitía pagar una licencia inicial de miles de euros. Además las modificaciones adicionales de *software* no estaban permitidas o incrementaban mucho el coste del proyecto.

- **Soluciones ERP SaaS (Cloud Computing):** Este tipo de solución era inicialmente mucho menos costosa que la solución anterior, pero el no

poder modificar las funcionalidades iniciales de forma sencilla ni barata fue el principal motivo de su descarte.

Tras el descarte de las anteriores soluciones, se comenzó la búsqueda de una solución de código abierto, se buscaba facilitar la personalización de la integración. A continuación se detallan las soluciones que se descartaron:

OpenERP: Se comenzó a hacer una selección de los más usados y los más actualizados, donde quedaron Openbravo, OpenERP y Dolibarr. Tras comparar los 3 ERP (Tabla 4) se descartó OpenERP por su mal soporte, su mala documentación y el lenguaje en el que estaba desarrollado.

OpenBravo: Al tener que elegir entre OpenBravo y Dolibarr, se comenzó a investigar por el *software* a priori más potente: Openbravo. Se comenzó configurando un servidor Tomcat para poder instalar Openbravo. Tras unos días, se consiguió tenerlo operativo. La tarea de instalación fue más compleja de lo esperado, ya que al estar desarrollado en Java, era necesario seguir una serie de pasos para preparar el entorno. Tras la instalación (compleja) se comenzó a realizar una serie de pruebas para poder extender las funcionalidades. Pero debido a una actualización de hace unos meses en el *software*, toda la información sobre desarrollo que se necesitaba estaba desactualizada. Se comprobó que el desarrollo sería lento. Y no se podrían cumplir los plazos, ya que la curva de aprendizaje era demasiado alta debido a la carencia de información. Además OpenBravo era un ERP de código abierto hasta su versión 2.5, pero en la versión 3.0 la documentación de desarrollo pasó a estar en manos de empresas que comercializan sus servicios, por lo que dificulta el perfeccionamiento de la plataforma por la comunidad de desarrolladores.

Una vez descartados estos sistemas, se comenzó con el desarrollo e implantación del ERP Dolibarr. Actualmente no existe ningún módulo para comunicar Magento y Dolibarr. Elegir Dolibarr para realizar la integración supondrá una ayuda para futuros proyectos de integración entre aplicaciones en toda la comunidad de desarrollo.

6.2. Búsqueda y selección de plataforma de comercio electrónico.

En segundo lugar, se procede a realizar un proceso de selección de una plataforma de comercio electrónico. Para realizar la búsqueda de una plataforma de eCommerce, se analizan las soluciones de código abierto más populares del mercado. Estas son actualmente las más usadas en los nuevos desarrollos de comercios *online:*

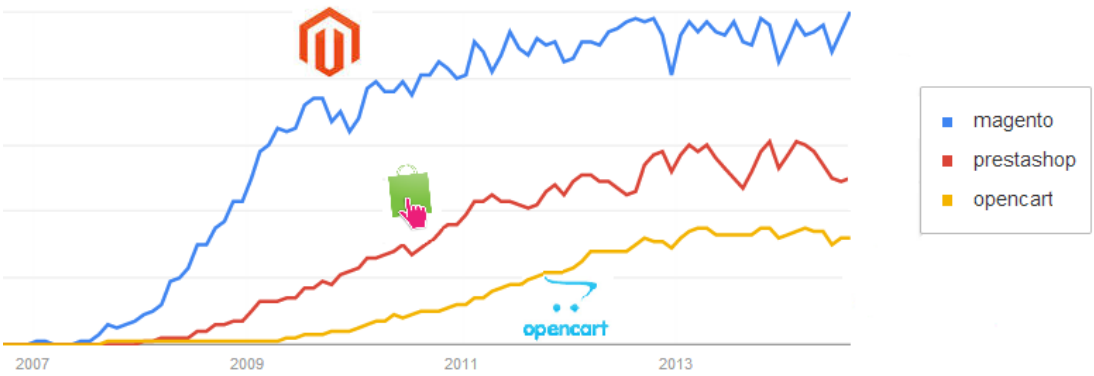


Ilustración 17: Plataformas de eCommerce más populares. GoogleTrends

Después de una primera comparación, se aprecia en la ilustración superior que la popularidad de OpenCart es pequeña. Además OpenCart está un paso por detrás en funcionalidades y rendimiento. A continuación se muestra la tabla de comparación entre Magento y Prestashop:

		Magento Open Source eCommerce	PRESTASHOP The Best E-Commerce Experience
Instalación e implantación	**Requisitos para la instalación**	• PHP junto a MySQL *FrameWork Zend*. • Versión Community (Gratis) • Versión Enterprise (12990$/año) Linux x86,x86-64.	• PHP 5 o superior. • FrameWork Smarty. • Servidor Web Apache. MSQL 5 o superior. • Servidor Linux, Unix o Windows.
	Implantación, Disponibilidad y Alojamiento.	Sí Cloud Computing. Servidor dedicado.	Sí. Cloud Computing.
Provee-dores	**Gestión de Proveedores**	Sí	Sí
	Gestión de Clientes	Sí	Sí
Pagos y facturación	**Gestión de Pagos**	Tarjeta de Crédito (VISA, MASTERCARD, AMEX, Discover), PayPal, Transferencia, Contra Reembolso.	PayPal, Hipay, Pago por cheque, Contra-Reembolso, Google CheckOut, etc.
	Facturas Electrónicas	Sí, extensión de pago.	No

Productos	**Gestión de Catálogo de Productos.**	Sí	Sí
	Gestión de Inventario	Sí	Sí
	Gestión Lotes de Productos	Sí	No
	Comparación de productos	Sí	No. Se puede comprar aparte
	Búsqueda de Productos	Sí	Sí
Carrito y lista de compra	**Gestión Carrito de compra.**	Sí	Sí
	Gestión Listas de Compra	Sí	No por defecto
Pedidos	**Gestión de Pedidos**	Sí	Sí
	Histórico de Pedidos	Sí	Sí
Logística	**Gestión de envíos y Distribución del Producto**	Sí. UPS, UPS XML, FedEx, USPS, SEUR, MRW y DHL	Sí. Permite selección de empresa transportista de envío por producto e incluso en función de rangos de importe.
Precios	**Gestión de Precios y Promociones**	Sí. Algunas funcionalidades requieren previo pago, como por ejemplo la visualización de Banners de Promociones, al comprar X productos.	Sí. Promociones o secciones del tipo: • Promociones especiales. • Los más vendidos. • Novedades.
	Múltiples Tipos de Impuestos	Sí	Sí
	Cupones Descuento	Sí	Sí

Redes sociales y comunicación	**Acceso a Red Social o Comunidad Web (Blogs, Eventos, Foros)**	Sí. • Permite crear Blogs, pero no eventos ni foros. • Posibilidad de integrar en Facebook (previo pago). • Falta incluir la accesibilidad mediante DNI-e	Sí. • Permite crear Blogs, foros (no eventos). • Permite integración con Facebook. • Falta incluir la accesibilidad mediante DNI-e.
	Comentarios en productos	Sí	Sí
	Valoración de productos	Sí	Sí
	NewsLetters	Sí	Sí
	Integración de Canales RSS	Sí	Sí
Funcionalidades	**Accesibilidad Mobile**	Sí. • Temas Responsive.	Si
	Emisión de informes estadísticos de indicadores	Sí. • Informes de Media de Pedidos. • Últimos 5 pedidos. • Últimas búsquedas de artículos. • Los productos más demandados. • Estadísticas muy completas sobre las ventas y tráfico de la plataforma *Web*.	Sí. • Estadísticas de importe de ventas realizadas. • Estadísticas de usuarios registros en tienda. • Estadísticas de pedidos finalizados. • Estadísticas de productos visitados. • Calendario de visitantes por fecha. • Permite mostrar estadísticas en forma de gráficos.
	Backup y Restauración BBDD	Sí	Sí
	Soporte Multi-idioma	Sí. No está incluida la traducción completa a lenguas autónomas. Las plantillas vienen en inglés.	Sí. Permite un sistema de traducciones por sección. Catalán, Inglés.

	Eventos	Sí. Desarrollo mediante módulos.	No a tiempo real
	Web Services	SI. XML-RPC,SOAP v1 SOAP v2, REST	SI REST (mal documentado)
Desarrollo	**Dificultad de Modificación**	Media	Fácil
	Soporta Multitienda:	Si	No
	Dificultad de Instalación	Media-Alta	Media-Fácil
Recomendado para:	**Empresas pequeñas:**	Si (ciertos casos)	Si
	Desarrolladores:	Si	Si
	Empresas medianas y grandes:	Si	No

Tabla 5: Comparativa Magento vs Prestashop

En conclusión, Prestashop es más fácil de usar y tiene menos requisitos hardware que Magento, el cual es más difícil de configurar y personalizar por primera vez.

Magento, por lo tanto, es un sitio de comercio electrónico pensado para empresas grandes o con un gran crecimiento. El poder realizar multitienda en Magento, es una característica muy importante para sitios de comercio electrónico que tienen mucho futuro. Mientras que Prestashop actualmente es usado por pequeños y medianos comerciantes.

Las funcionalidades entre las dos plataformas son bastante similares, sobre todo debido a la existencia de módulos complementarios en ambas plataformas. No obstante, a la hora de gestionar un comercio electrónico, Magento es más completo que Prestashop. Pero a su vez Magento es más difícil de gestionar por una sola persona debido al elevado número de funcionalidades, por lo que Magento requiere una formación mayor para el usuario administrador.

El desarrollo de módulos es más complejo en Magento debido a su arquitectura. Magento al estar basado en el *framework Zen,* hereda la forma de desarrollo de este, que requiere documentarse para realizar cada cambio. Conocer *PHP* no implica poder desarrollar en Magento

La forma de proceder para la elección de una plataforma de comercio electrónico fue similar a la del ERP. Inicialmente se barajaba la posibilidad de una solución propietaria, pero estas soluciones eran caras debido a las licencias. O en su defecto, su código no estaba sujeto a modificaciones. Por lo que analizando las soluciones de código abierto, la cuestión era Magento o Prestashop.

El descarte fue Prestashop. Se vio que la API de Magento era mucho más extensa, la API proporcionaba los métodos suficientes para poder realizar la integración. No obstante se continuó buscando diferencias. Magento permitía usar tanto SOAP como REST, mientras que Prestashop solo REST. A la hora de implementar un nuevo Servicio Web en Prestashop había que hacer uso de bibliotecas de código no oficiales, las cuales estaban sin actualizar desde hace varios años. Mientras que la documentación de Magento y sus bibliotecas de código eran recientes. La razón que determinó la selección de Magento, fue la mayor posibilidad de desarrollar módulos para integración, ya que el núcleo del desarrollo es la comunicación entre aplicaciones.

6.3. Magento.

Una vez finalizado el análisis de las distintas plataformas de comercio electrónico, y seleccionado Magento como plataforma para implantar una tienda *online*. Se procede a describir esta plataforma en detalle.

Ilustración 18: Logo Magento

Magento fue creada en 2007, está basada en Zen *framework* y su código fuente está escrito en lenguaje PHP. Está diseñado para ser instalada en un servidor *Web* en el cual se requiere una base de datos para completar su funcionamiento. Actualmente se ha consolidado como la plataforma más popular de eCommerce en el mundo [13]. Que esta plataforma sea la solución más usada no es ninguna casualidad, ya que posee unas características que lo diferencian de sus competidores. Las más importantes son:

- Funcionalidades iniciales: Magento viene con una serie de funcionalidades listas para su uso tras su instalación. Con su *BackOffice* se pueden realizar múltiples acciones esenciales para llevar adecuadamente el comercio *online*. Por ejemplo, la gestión total del catálogo, la optimización de búsqueda, analítica *Web*, reportes, gestión de cobros y envíos, administración de usuarios o integración con programas de facturación/gestión…

- Comunidad y Soporte: Magento es una plataforma de código abierto. Su código puede ser modificado por los desarrolladores una vez instalado. Gracias a ello, tiene un grupo de desarrollo detrás, que se ha encargado de formar una solución cada día más grande que da como resultado un sistema robusto, flexible y escalable. La comunidad, ha logrado que el sistema tenga menos errores, gracias a personas que se encuentran vigilando y arreglado posibles errores, además de generar y producir nuevas ideas para futuras versiones.

- Principales Características:

  - **Robusto**. Se puede decir que se está ante un sistema muy seguro. Tras 7 años se ha logrado un sistema sin fallos de seguridad. Magento es una plataforma alojada en la nube que necesita unos

servidores más equipados que otras plataformas como Prestashop. Para ello es recomendable que Magento esté alojado en servidores dedicados o VPS (servidor virtual privado), ya que así se puede conseguir un buen rendimiento. Actualmente se encuentra en la versión 1.9, donde ya se encuentran solucionados todos los errores que se habían detectado en las versiones anteriores.

- **Flexible.** Su diseño es una de las cosas que más puede sorprender. Magento puede ser personalizado al milímetro, puede adaptarse tanto a tiendas B2B (*business to business*), como B2C (*business to client*), incluso soportar múltiples tiendas en una única instalación, teniendo centralizado en un solo lugar varios comercios. Debido a la importancia del comercio desde los dispositivos móviles, Magento adoptó un diseño adaptativo a resoluciones más pequeñas.

- **Escalable**. Existe una multitud de extensiones para Magento, las cuales amplían las funciones básicas. Con las extensiones se puede: integrar la plataforma con las redes sociales, conectarlo con un ERP, CRM, empresa de logística, realizar configuraciones especiales o mejorar características ya implementadas como la exportación e importación.

- Localización e Internacionalización.

Debido al abanico de posibilidades que tiene Magento, es posible realizar una tienda pensada para vender en todos los lugares del mundo, teniendo tanto la capacidad de adaptarse a varios idiomas, detectando el lugar de cada cliente de la tienda, como de poder gestionar los impuestos correspondientes al país desde el que se realiza la compra e incluso se pueden llegar a configurar distintos tipos de envío en función del país destinatario.

6.4. Dolibarr

Dolibarr ha sido el ERP que más se ha adecuado a las necesidades de implantación del caso de estudio. Dolibarr es un *software* de código abierto para la gestión de PYMES y organizaciones. Es a la vez un ERP y un CRM que se aloja en un servidor *Web* [14].

Ilustración 19: Logo Dolibarr

Las funcionalidades básicas de Dolibarr son:

- Gestión de productos y servicios: proporciona una serie de tablas y formularios para realizar la tarea de administrar productos.
- Gestión comercial, se realiza un seguimiento comercial para clientes y pedidos. Esta funcionalidad es típica de un CRM.
- Gestión de usuarios, se pueden gestionar tanto usuarios clientes de la empresa, como proveedores o trabajadores internos.

- Gestión financiera, en esta sección se pueden centralizar, balances, facturas y demás procesos financieros.
- Gestión de documentos, se pueden gestionar PDF (portable document format), catálogos o archivos de tipo CSV (comma-separated values) de importación/exportación.
- Módulo de terminal de venta (TPV), módulo para ser usado por el vendedor de las tiendas físicas. Proporciona las funcionalidades necesarias para poder realizar pagos en metálico, tarjeta o transferencia en una tienda física.
- Gestión de contratos, se pueden tener clasificados todos los contratos de la empresa en una serie de tablas.

Estas funcionalidades son complementadas en el Anexo 5: Funcionalidades Dolibarr, donde se ofrece una tabla con los distintos servicios ofrecidos por el ERP Dolibarr.

Dolibarr está desarrollado en PHP. Modificar el funcionamiento o crear nuevas funcionalidades, requieren conocer la arquitectura de Dolibarr. La estructura está dispuesta de forma modular, cada funcionalidad es implementada por un módulo distinto, lo que lo hace más ágil. El desarrollo de un módulo se verá más en detalle en el apartado 8.1 Módulo Dolibarr. Toda la información es almacenada en una base de Datos MySQL.

Los módulos externos más usados en Dolibarr, con los que dejar el sistema completo, son los siguientes:

- **Módulo Multicompany**: permite trabajar con varias empresas en una sola implantación de Dolibarr
- **Módulo 2Reports**: con esta extensión se pueden generar informes en PDF, HTML y CSV.
- **Módulo Google**: este módulo proporciona acceso al API de google: contactos, calendario, etc.
- **Módulo MemCache**: proporciona una caché que incrementa la velocidad de uso de Dolibarr.
- **Módulo DoliPOS**: punto de venta TPV profesional para Dolibarr.
- **Módulo Paypal**: integración de Paypal con Dolibarr.
- **Módulo ConcatPDF**: fusión de PDF con presupuestos, pedidos y facturas.
- **Módulo Dolipresta**: integración de Dolibarr con la gestión de pedidos de Prestashop.
- **Módulo 2Webmail**: Cliente de correo para Dolibarr.

A continuación se detalla en forma de grafico la arquitectura interna del ERP.

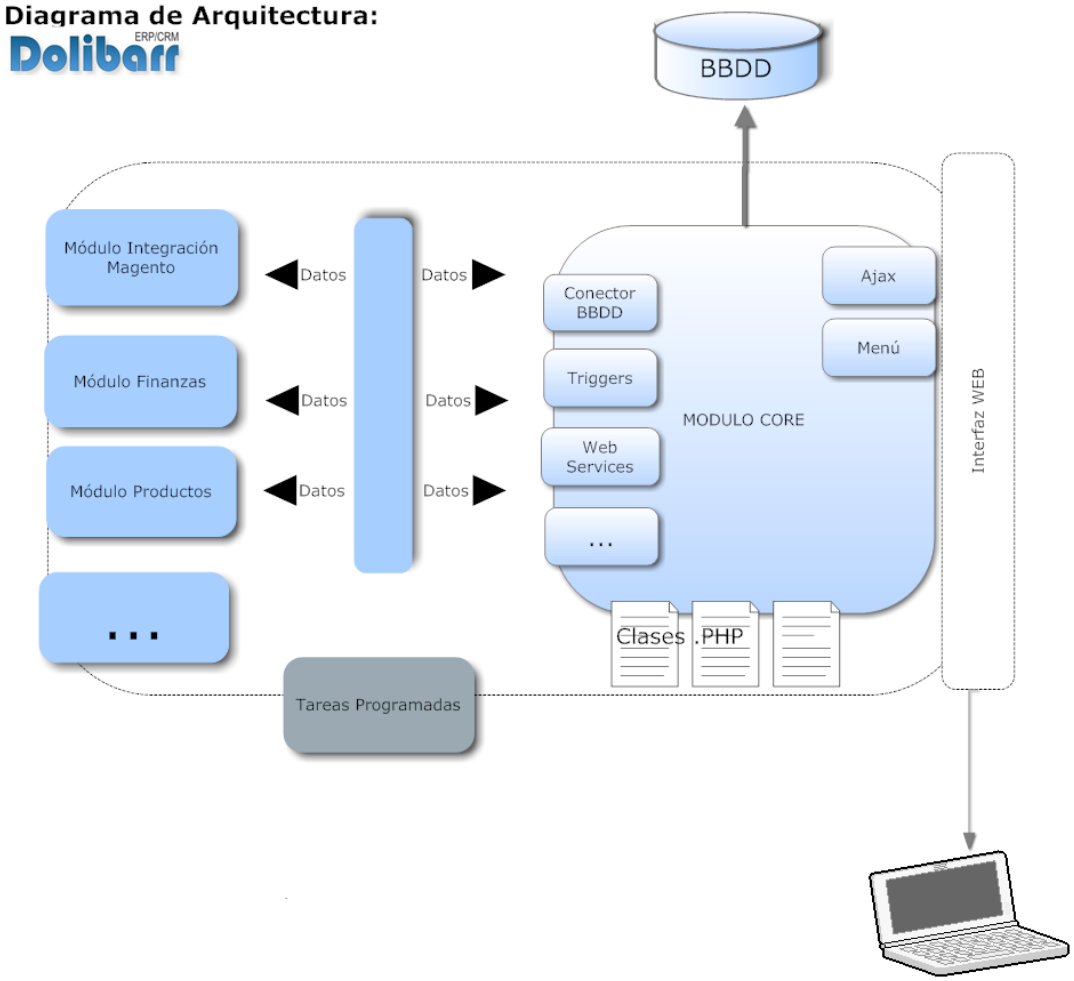


Ilustración 20: Arquitectura de la disposición de Dolibarr

Dolibarr se encuentra creciendo con paso firme. Algunos de los motivos por los que está progresando son: el creciente interés mostrado por las empresas, provocado por la satisfacción de los clientes que ya lo tienen implantado. También está progresando, gracias a que cada vez se añaden más idiomas a la interfaz, por lo que se está accediendo a nuevos países. Últimamente se han añadido los siguientes idiomas: griego, árabe, italiano, chino y japonés. Varias empresas en todo el mundo están colaborando en su desarrollo, creando poco a poco un gran sistema y prometiendo un gran futuro.

Capítulo 7. Aplicación de la metodología al caso de estudio.

A continuación se procede a la **validación** de la metodología elaborada en el Capítulo 5, con un caso de estudio determinado, el cual se detalla en el Capítulo 4.

En primer lugar, la metodología propuesta indica que se deben definir los requisitos del sistema, incluyendo tanto los funcionales como los no funcionales.

En el siguiente apartado se detallan los diferentes requisitos que serán implementados en la solución final. La elaboración de un proyecto de desarrollo e integración necesita unos requisitos para mejorar la comprensión del alcance de la solución. Se utilizan dos conjuntos de requisitos:

- **Requisitos funcionales:** son los requerimientos que definen una función del sistema. Definen acciones que han de estar implantadas en la solución desarrollada.

- **Requisitos no funcionales:** son los requerimientos que establecen ciertas limitaciones o condiciones en aspectos de calidad, costes, estabilidad, estándares y portabilidad.

7.1.1. Requisitos funcionales

A continuación se detallan los requisitos funcionales del sistema:

Identificador	**RF-001**		
Nombre	Inventario actualizado general.		
Descripción	Debe tener el inventario de los productos actualizado entre ERP y eCommerce en tiempo real.		
Prioridad	Obligatorio	Dificultad	Alto

Tabla 6: RF-001. Inventario actualizado general

Identificador	RF-002		
Nombre	Inventario actualizado en eCommerce		
Descripción	Debe mostrar a la venta en la tienda *online*, solo productos en stock. Para ello es necesario realizar cambios inmediatamente entre aplicaciones.		
Prioridad	Obligatorio	**Dificultad**	Alto

Tabla 7: RF-002. Inventario actualizado en eCommerce

Identificador	RF-003		
Nombre	Terminal Punto de Venta		
Descripción	Se requiere un terminal (TPV) punto de venta para las tiendas físicas.		
Prioridad	Obligatorio	**Dificultad**	Medio

Tabla 8: RF-003. Terminal punto de venta

Identificador	RF-004		
Nombre	Múltiple TPV		
Descripción	Tener un TPV que pueda estar implantado en distintas tiendas físicas, trabajando todos los TPV concurrentemente.		
Prioridad	Obligatorio	**Dificultad**	Medio

Tabla 9: RF-004. Múltiple TPV

Identificador	RF-005		
Nombre	Sincronización inmediata		
Descripción	Actualizar la información de manera asíncrona cada vez que se produzca un cambio.		
Prioridad	Obligatorio	**Dificultad**	Alto

Tabla 10: RF-005. Sincronización inmediata

Identificador	RF-006		
Nombre	Sincronización diaria		
Descripción	Debe ejecutar una sincronización diaria que verifique y corrija los cambios de inventario.		
Prioridad	Opcional	**Dificultad**	Alto

Tabla 11: RF-006. Sincronización diaria

Identificador	RF-007		
Nombre	Movimientos de almacén		
Descripción	Cuando se produzca un movimiento en el almacén, tanto de entrada como de salida. Es necesario que se modifique el inventario en todo el sistema		
Prioridad	Obligatoria	**Dificultad**	Medio

Tabla 12: RF-007. Movimientos de almacén

Identificador	RF-008		
Nombre	Modificación productos		
Descripción	Se ha de poder realizar cambios en los productos y que estos cambios se propaguen en todo el sistema.		
Prioridad	Opcional	Dificultad	Medio

Tabla 13: RF-008. Modificación productos

Identificador	RF-009		
Nombre	Creación producto		
Descripción	Cuando se crea un producto en el ERP, la BBDD (base de datos) de la tienda *online* debe ser actualizada.		
Prioridad	Obligatorio	Dificultad	Alto

Tabla 14: RF-009. Creación producto

Identificador	RF-010		
Nombre	Importación masiva productos		
Descripción	Cuando se realiza una importación masiva de productos desde un archivo de tipo .csv o de tipo. Xml es necesario mantener la sincronización en el sistema.		
Prioridad	Opcional	Dificultad	Alto

Tabla 15: RF-010. Importación masiva productos

7.1.2. Requisitos no funcionales

A continuación se detallan los requisitos no funcionales del sistema:

Identificador	RNF-001		
Nombre	Conexión		
Descripción	Se requiere una conexión a Internet para acceder al servidor y a las aplicaciones.		
Prioridad	Obligatoria	Dificultad	Baja

Tabla 16: RNF-001. Conexión

Identificador	RNF-002		
Nombre	Implantación plataforma		
Descripción	Debe Implantar un ERP y eCommerce que pueda ser accesible desde cualquier dispositivo.		
Prioridad	Obligatorio	Dificultad	Alto

Tabla 17: RNF-002. Implantación plataforma

Identificador	RNF-003		
Nombre	Referencia global		
Descripción	La forma de identificar a los distintos productos ha de ser por el número de referencia del producto (nunca por ID), se necesita que la referencia sea única.		
Prioridad	Obligatoria	Dificultad	Baja

Tabla 18: RNF-003. Referencia global

Identificador	RNF-004		
Nombre	Retardo integración		
Descripción	El retardo en las llamadas a Servicios Web debe ser menor de 5 segundos.		
Prioridad	Obligatoria	Dificultad	Media

Tabla 19: RNF-004. Retardo integración

Identificador	RNF-005		
Nombre	Existencias Stock		
Descripción	Si no existe inventario en el ERP, la tienda *Web* no debe permitir vender ningún producto.		
Prioridad	Obligatoria	Dificultad	Media

Tabla 20: RNF-005. Existencias Stock

Identificador	RNF-006		
Nombre	Creación productos nuevos.		
Descripción	Solo se podrán crear productos en el ERP.		
Prioridad	Obligatoria	Dificultad	Baja

Tabla 21: RNF-006. Creación productos nuevos

Identificador	RNF-007		
Nombre	Servidor		
Descripción	Necesidad de servidor para el alojamiento de las aplicaciones, de al menos 8GB de memoria RAM, 100 GB de disco duro, 100Mbps de conectividad a Internet, con la disponibilidad de al menos dos direcciones IP para el servidor.		
Prioridad	Obligatoria	Dificultad	Baja

Tabla 22: RNF-007. Servidor

Identificador	RF-008		
Nombre	Integración transparente		
Descripción	Debe existir una integración transparente para el usuario, no puede haber retardos excesivos en la comunicación.		
Prioridad	Obligatorio	Dificultad	Medio

Tabla 23: RNF-008. Integración transparente

7.2. Casos de uso

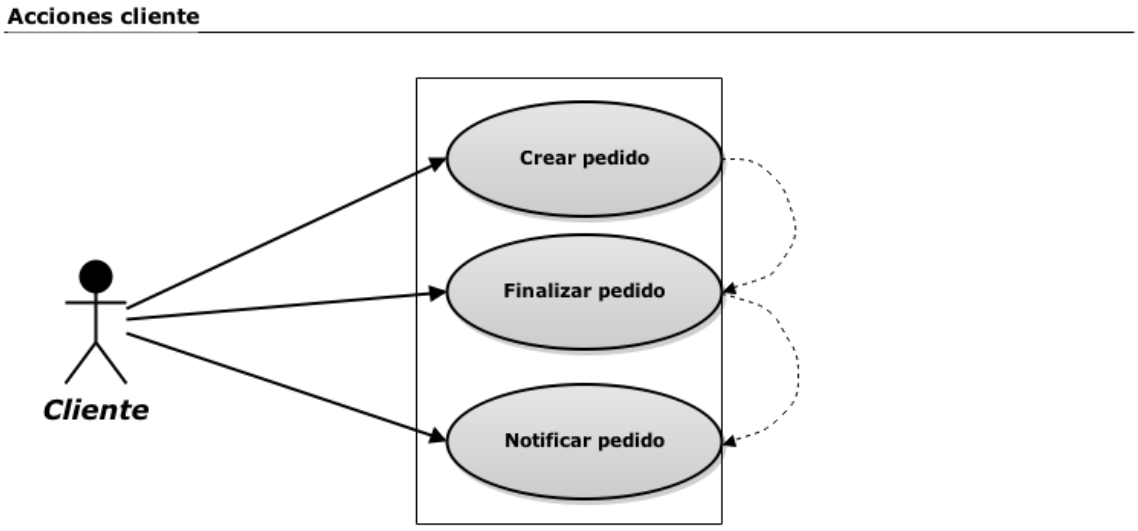
El segundo paso de la metodología propuesta, consiste en definir los casos de uso. Estos son los posibles escenarios que se pueden dar en el sistema, llevados

a cabo por algún usuario [15]. Estas situaciones son ilustradas mediante diagramas que ayudan en el desarrollo de los módulos de Dolibarr y Magento.

Seguidamente se listan los diagramas de caso de uso de todos los actores de la solución diseñada.

7.2.1. Cliente tienda online.

Se comienza detallando el diagrama de caso de uso del cliente de la tienda *online*. En este apartado se describen las acciones del cliente que influyen en el proyecto, pudiendo realizar multitud de tareas de administración independientes del actual caso de estudio.



Las descripciones de los distintos casos de uso correspondientes al cliente son:

- Caso de uso: Crear pedido

Identificador	CL-1
Nombre	Crear Pedido
Actor	Cliente
Objetivo	Crear un pedido en el que se incluyan las cantidades, nombres y precios de los productos. Para poder procesar el pedido posteriormente.
Precondiciones	N / A
Postcondiciones	N / A
Escenario básico	-El cliente va añadiendo productos al carrito según necesite. -Para añadir al carrito debe comprobar que los ítems tienen stock disponible. -Cuando se completa la compra se procede a finalizar el pedido.

Tabla 24: Descripción del caso de uso CL-1: Crear pedido

- Caso de uso: Finalizar Pedido

Identificador	CL-2
Nombre	Finalizar Pedido
Actor	Cliente
Objetivo	Realizar el pago y añadir las direcciones de entrega y facturación.
Precondiciones	Haber creado un pedido.
Postcondiciones	Quedar registrado en la tienda.
Escenario básico	-El cliente con el pedido ya creado, procede a indicar la información completa del pago. -El formulario de pago requiere información de envío, tipo de envío, información de facturación. -El cliente selecciona e introduce los datos de pago (transferencia, pago con tarjeta…).

Tabla 25: Descripción del caso de uso CL-2: Finalizar pedido

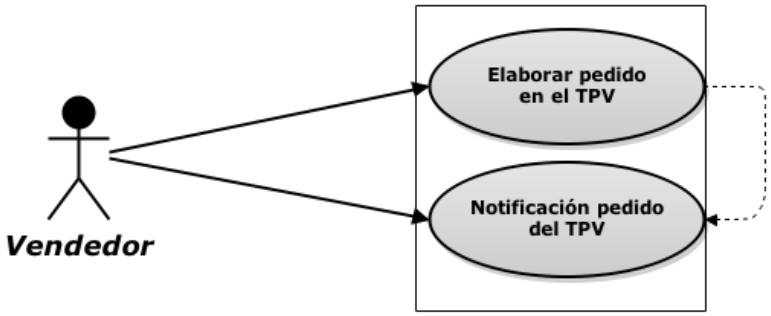
- Caso de uso: Notificar Pedido

Identificador	CL-3
Nombre	Notificar Pedido
Actor/es	Cliente/Magento/Dolibarr
Objetivo	Remitir la información del nuevo pedido que acaba de finalizar el cliente.
Precondiciones	Realizar un pedido exitosamente.
Postcondiciones	N / A
Escenario básico	-El cliente al indicar la finalización de pedido provoca la ejecución de un evento interno en la tienda *online*. -El evento notificará los cambios de inventario al nodo central (Dolibarr-ERP).

Tabla 26: Descripción del caso de uso CL-3: Notificar Pedido

7.2.2. Vendedor Dolibarr-ERP.

Seguidamente se detalla el diagrama de caso de uso del vendedor, la persona encargada de gestionar una tienda desde el TPV de la tienda física. También se enumeran las descripciones de los casos de uso del vendedor.



Las descripciones de los distintos casos de uso correspondientes al vendedor son:

- Caso de uso: Elaborar pedido en el TPV.

Identificador	V-1
Nombre	Elaborar pedido en el TPV.
Actor/es	Vendedor
Objetivo	Realizar un pedido con los productos indicados por el cliente mediante el TPV (extensión de Dolibarr).
Precondiciones	Autenticarse en el TPV. Realizar el pedido en la tienda física.
Postcondiciones	N / A
Escenario básico	- El cliente selecciona una serie de productos en la tienda. -El Vendedor los añade a la lista de compra. -Finalizar el pedido y elegir la modalidad de pago. -El cliente paga y se le emite un ticket.

Tabla 27: Descripción del caso de uso V-1: Elaborar pedido en el TPV.

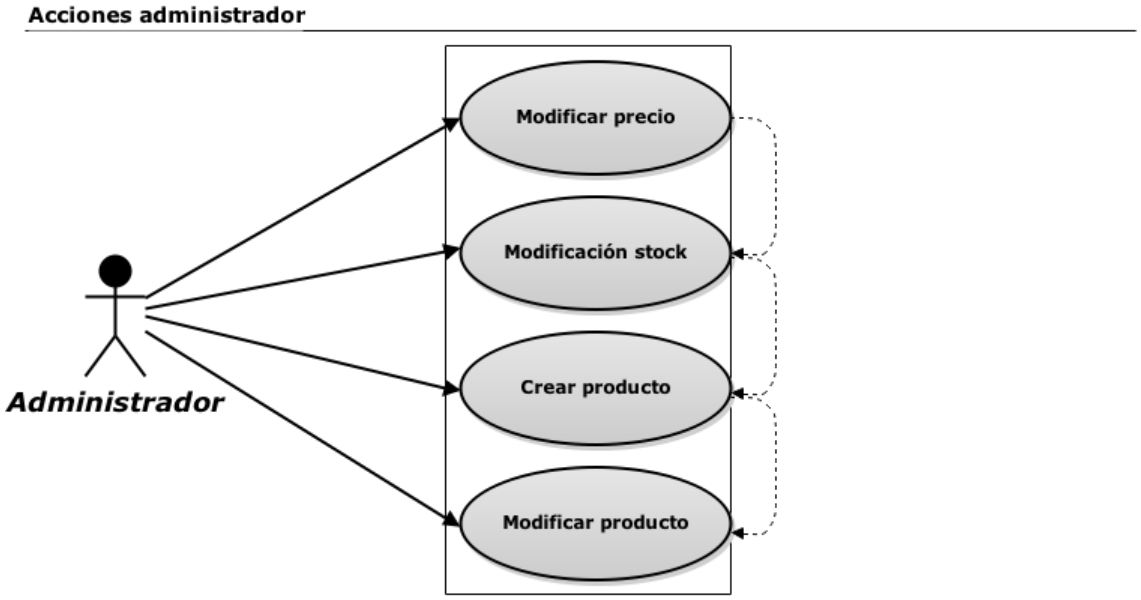
- Caso de uso: Notificar pedido del TPV.

Identificador	V-2
Nombre	Notificar pedido del TPV.
Actor/es	Vendedor
Objetivo	Notificar al ERP y a la tienda *online* sobre el cambio de inventario en los productos del pedido.
Precondiciones	-Finalización del pedido exitosamente. -Los productos deben estar en stock.
Postcondiciones	N / A
Escenario básico	-El terminal (TPV) de Dolibarr provoca un evento de fin de pedido. -Al saltar el evento, se invoca un método que le comunica a la base de datos de la tienda *online* que el stock de ciertos productos ha cambiado.

Tabla 28: Descripción del caso de uso V-2: Notificar pedido TPV

7.2.3. Administrador ERP.

A continuación se detalla el diagrama de caso de uso del administrador del ERP, la persona encargada de gestionar el ERP Dolibarr.



Las descripciones de los distintos casos de uso correspondientes al administrador son:

- Caso de uso: Modificar Precio.

Identificador	AERP-1
Nombre	Modificar Precio
Actor/es	Administrador
Objetivo	Corrección precio en un producto.
Precondiciones	El producto que se quiere modificar ha de estar creado en la BBDD del ERP.
Postcondiciones	N / A
Escenario básico	-Acceder a la pestaña Productos. -Seleccionar producto. -Acceder a la sub-pestaña "precios clientes". -Cambiar el campo en el formulario "precio de venta". -Grabar la operación.

Tabla 29: Descripción del caso de uso AERP-1: Modificar Precio

- Caso de uso: Modificar Stock.

Identificador	AERP-2
Nombre	Modificar Stock
Actor/es	Administrador ERP
Objetivo	Realizar movimiento de inventario tanto de entrada en el almacén, como salida del almacén.
Precondiciones	El producto que se quiere modificar ha de estar creado en la BBDD del ERP.
Postcondiciones	N / A
Escenario básico	-Acceder a la pestaña Productos. -Seleccionar producto. -Acceder a la sub-pestaña "Stock". -Pasar a modo "corrección stock". -Realizar el movimiento de inventario en el almacén. -Grabar la operación.

Tabla 30: Descripción del caso de uso AERP-2: Modificar Stock.

- Caso de uso: Crear producto.

Identificador	AERP-3
Nombre	Crear producto
Actor/es	Administrador ERP
Objetivo	Añadir productos a la base de datos del ERP y a la tienda *online* simultáneamente
Precondiciones	-Tener un almacén creado en el ERP. -Tener nuevos productos sin introducir en el sistema.
Postcondiciones	-Para completar la creación de un producto es necesario añadir el inventario inicial en la pestaña "Stock" y las imágenes del producto en la pestaña "Fotos".
Escenario básico	- Seleccionar "nuevo producto" en la pestaña Productos - Rellenar los campos del formulario y se crea. - Rellenar adicionalmente el stock y las imágenes del producto en las pestañas correspondientes.

Tabla 31: Descripción del caso de uso AERP-3: Crear producto.

- Caso de uso: Modificar producto.

Identificador	AERP-4
Nombre	Modificar Producto
Actor/es	Administrador
Objetivo	Corrección datos en un producto.
Precondiciones	El producto que se quiere modificar ha de estar creado en la BBDD del ERP.
Postcondiciones	N / A
Escenario básico	-Acceder a la pestaña Productos →Ficha. -Acceder al modo "modificar". -Editar los campos(los marcados con el logo de Magento). -Grabar la operación.

Tabla 32: Descripción del caso de uso AERP-4: Modificar producto

De los requerimientos anteriores se extrae la información para elaborar un plan de diseño. Este está compuesto por un diseño de la topología del sistema, un esquema de sincronización y un diseño de la arquitectura del sistema junto a sus elementos.

7.3. Diseño de la solución

La metodología indica que a continuación se debe seleccionar una plataforma de comercio electrónico. Como se ha detallado en el apartado 6.2 del capítulo anterior, donde se comparan las distintas plataformas de comercio electrónico, se ha decidido seleccionar Magento entre las demás plataformas analizadas. Debido a que Magento es una solución manejable (de código abierto), fiable (tiene un gran reconocimiento por su robustez) y además permite realizar el caso de estudio mediante la creación de un módulo para la integración.

Para el ERP, se ha decidido usar Dolibarr, esta decisión se justifica en el apartado 6.1 del capítulo anterior. Dolibarr, al ser de código abierto también permite llevar a cabo el desarrollo del caso de estudio. Además, al incorporar un módulo de Servicios Web, se puede utilizar la estructura de este módulo, para desarrollar la integración que se tiene como objetivo.

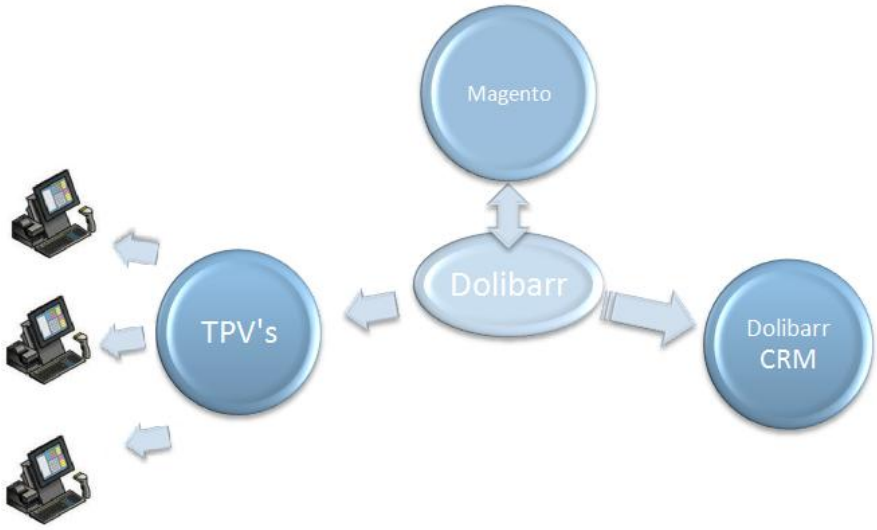


Ilustración 21: Topología real del sistema.

De acuerdo con lo sugerido en la metodología. Para diseñar la estructura de la integración, se ha elegido una topología de integración centralizada (*Hub & Spoke*), ya que en este proyecto se van a integrar dos sistemas: Magento y Dolibarr. Los TPV que son un elemento más de la topología, son implantados mediante un módulo que ya está integrado por defecto en Dolibarr. No se ha elegido una topología *bus,* ya que no se iba a utilizar la ventaja de añadir sistemas adicionales a la solución.

Para la integración de los distintos datos en la aplicación es necesario elaborar un bosquejo con los datos que se quieren tener sincronizados en ambos sistemas. Para ello se elabora el siguiente esquema:

	ERP →eCommerce	**eCommerce→ERP**
Productos nuevos	✓ Si	× No
Modificar Productos	✓ Si	× No
Estado Stock	✓ Si	✓ Si
Movimientos Almacén	✓ Si	× No
Pedidos/Ventas	✓ Si (TPV)	✓ Si

Tabla 33: Esquema de sincronización real.

A continuación, siguiendo la imagen superior, se describe la forma en que se despliega la sincronización para cada conjunto de datos.

- Los **nuevos productos** de la empresa que se introduzcan en el sistema se crearán desde Dolibarr. Seguidamente mediante Servicios Web se añaden de forma remota los productos en la BBDD de Magento.

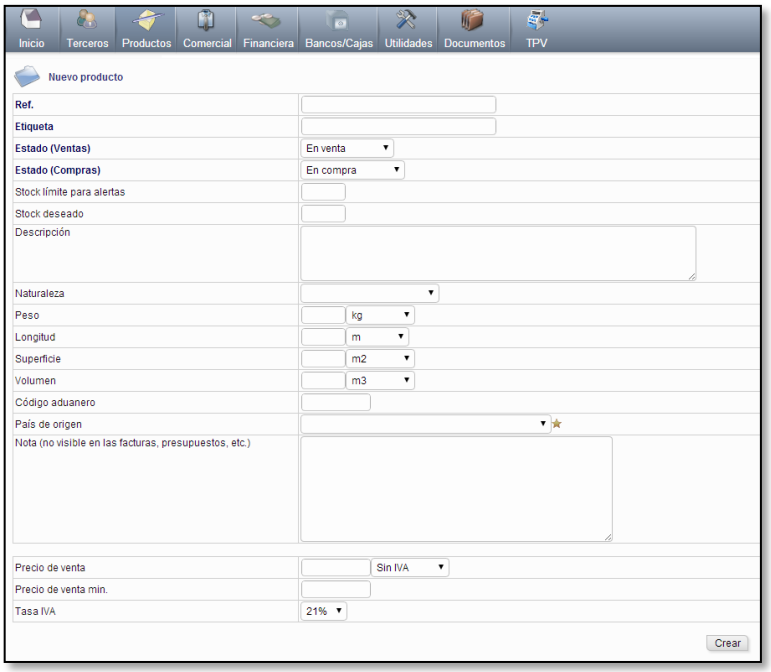


Ilustración 22: Formulario creación nuevos productos.

- Los productos pueden ser **modificados** en el ERP en la sección: Productos → Ficha → Modificar. Una vez modificados el sistema detecta que se ha producido un cambio y actualiza los campos que describen al producto (nombre, precio, descripción, peso...).

- El **stock** va a ser el campo que se quiere tener siempre actualizado, dado que es crítico que este siempre sea el correcto. Para ello se utilizan los eventos de Magento, de Dolibarr y los Servicios web. Además la sincronización requiere de un identificador común en ambas aplicaciones, el **número de referencia**. Una vez creado el producto no puede ser editado, ya que existe una tabla tanto en Magento como en Dolibarr que vincula cada producto con su referencia.

- Los **movimientos** que se produzcan en el almacén, tanto de entrada como de salida, se capturan en el ERP y se modifica el nuevo estado del inventario en la tienda *online*.

- Cuando se produce una **venta** en la tienda *Online*. Mediante Servicios Web se comunica la cantidad vendida y la referencia del producto vendido. A su vez, la venta desde cualquier TPV produce un cambio en la base de datos de Dolibarr. Este cambio, notifica mediante Servicios web a Magento, para que actualice la cantidad vendida y su número de referencia.

Una vez definidos los datos a integrar, se pasa a comentar el flujo que se ha diseñado para la solución completa, incluyendo tienda *online* (Magento), ERP (Dolibarr) y los TPV (módulo Dolibarr).

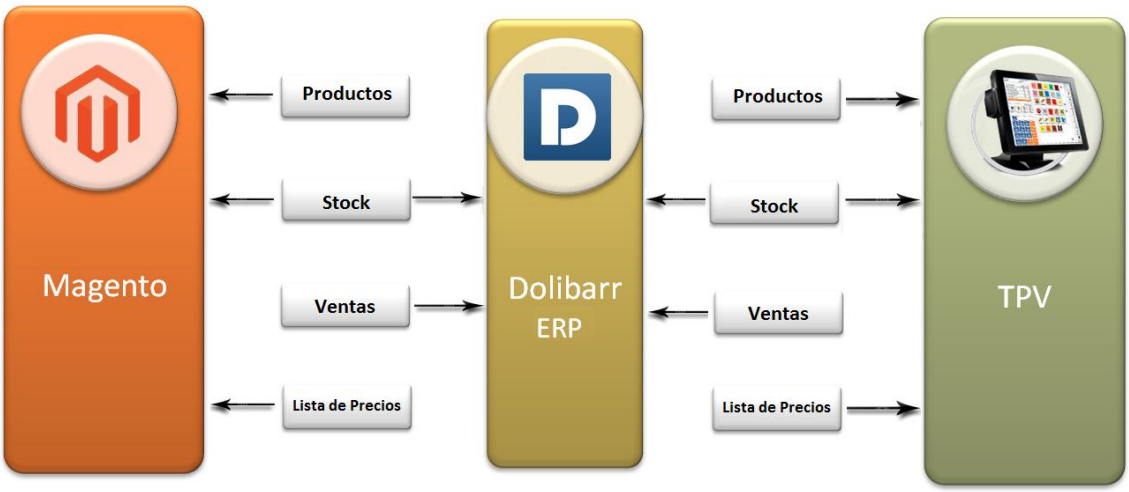


Ilustración 23: Esquema de funcionamiento del sistema.

En la ilustración anterior se muestran los mensajes que se intercambiarán cuando la solución esté desarrollada al final del actual proyecto. Este esquema complementa al esquema de sincronización de la *Tabla 33: Esquema de sincronización real.* En la ilustración se puede ver el flujo de datos necesario. El objetivo básico de la integración, como se puede apreciar, es el stock. Es por ello que las operaciones de la ilustración superior son acciones que permiten tener el inventario siempre actualizado.

También, la ilustración superior refleja la creación de un nuevo producto desde Dolibarr. Esto provoca que el producto esté disponible en la tienda *online* y en los TPV de las tiendas físicas. A su vez, cada venta realizada en cualquiera de los comercios, tanto en el *online* como en el físico, cambia el *stock* actual de la base de datos de Dolibarr mediante una notificación.

7.3.1. Arquitectura General

El diseño de la arquitectura final del sistema se encuentra representado en la Ilustración 24. En ella se diferencian dos elementos básicos: en la parte superior se ve el sitio de comercio electrónico implementado con la plataforma Magento, y en la parte inferior, el ERP Dolibarr con sus módulos y extensiones. Un aspecto muy importante que se representa en el diagrama de la Ilustración 24, es la comunicación que ha de existir entre el módulo del ERP Dolibarr y Magento. El sistema final, siguiendo la Ilustración 24 en orden descendente, quedaría de la siguiente manera:

En primer lugar se encuentran los usuarios que acceden al portal de comercio electrónico desde su navegador. Existen dos tipos: cliente y administrador, que

hacen uso del sistema de la plataforma de comercio electrónico, Magento.

Magento tiene dos elementos que componen la arquitectura de la plataforma de comercio electrónico [16]:

Por un lado el **núcleo,** que contiene las funcionalidades básicas para el correcto funcionamiento de la plataforma, tales como gestión de recurso, la interfaz de administración básica, la gestión de la base de datos (MYSQL), etc. Estas funcionalidades del "núcleo del código" son un conjunto de módulos desarrollados por el equipo oficial de desarrollo de Magento. A su vez, el núcleo está compuesto de algunas extensiones desarrolladas por empresas certificadas que se integran con las demás funcionalidades del núcleo.

Por otro lado, Magento tiene las **extensiones.** Que son herramientas adicionales que extienden la funcionalidad de Magento. Existen tres tipos:

- Los **Temas** están formados por archivos de diseño, plantillas y/o archivos de máscara que crean el aspecto visual de la tienda.

- Los **Bloques,** son los encargados de estructurar visualmente la tienda *online* (cabecera, barras laterales, etc.).

- Los **Módulos** aportan nuevas herramientas, como pueden ser la integración con pasarelas de pago, herramientas de marketing o módulos de comunicación con otros sistemas.

En el caso de la solución del actual caso de estudio, se implementa un módulo en Magento que añade la funcionalidad de capturar eventos, procesar la información y mandarla a otra aplicación (Servicios web), en este caso a Dolibarr.

La comunicación entre Magento y Dolibarr se realiza mediante Servicios Web. Los Servicios Web proporcionan una interfaz que puede ser llamada desde cualquier aplicación. Consiguiendo el requisito de la interoperabilidad entre aplicaciones.

Ambas aplicaciones se intercambian mensajes SOAP utilizando Internet como modo de envío. Estos mensajes son generados por Magento cuando actúa como cliente en el caso de producirse una venta. También son generados por Dolibarr en el caso de que se produzcan cambios en el ERP y sean notificados hacia Magento.

En el caso de Dolibarr, los Servicios Web funcionan de forma similar, han de estar implementados en un módulo externo al núcleo del sistema. Estos se comportan como cliente en distintos casos de uso, por ejemplo cuando se crea un nuevo producto en el sistema, cuando se modifica el stock, cuando se cambia el nombre a un producto, etc.

La interfaz de Dolibarr es más sencilla como se puede apreciar en la ilustración 25. La base de datos es MySQL. El módulo del TPV, será utilizado por todas las tiendas físicas especificadas en el caso de estudio.

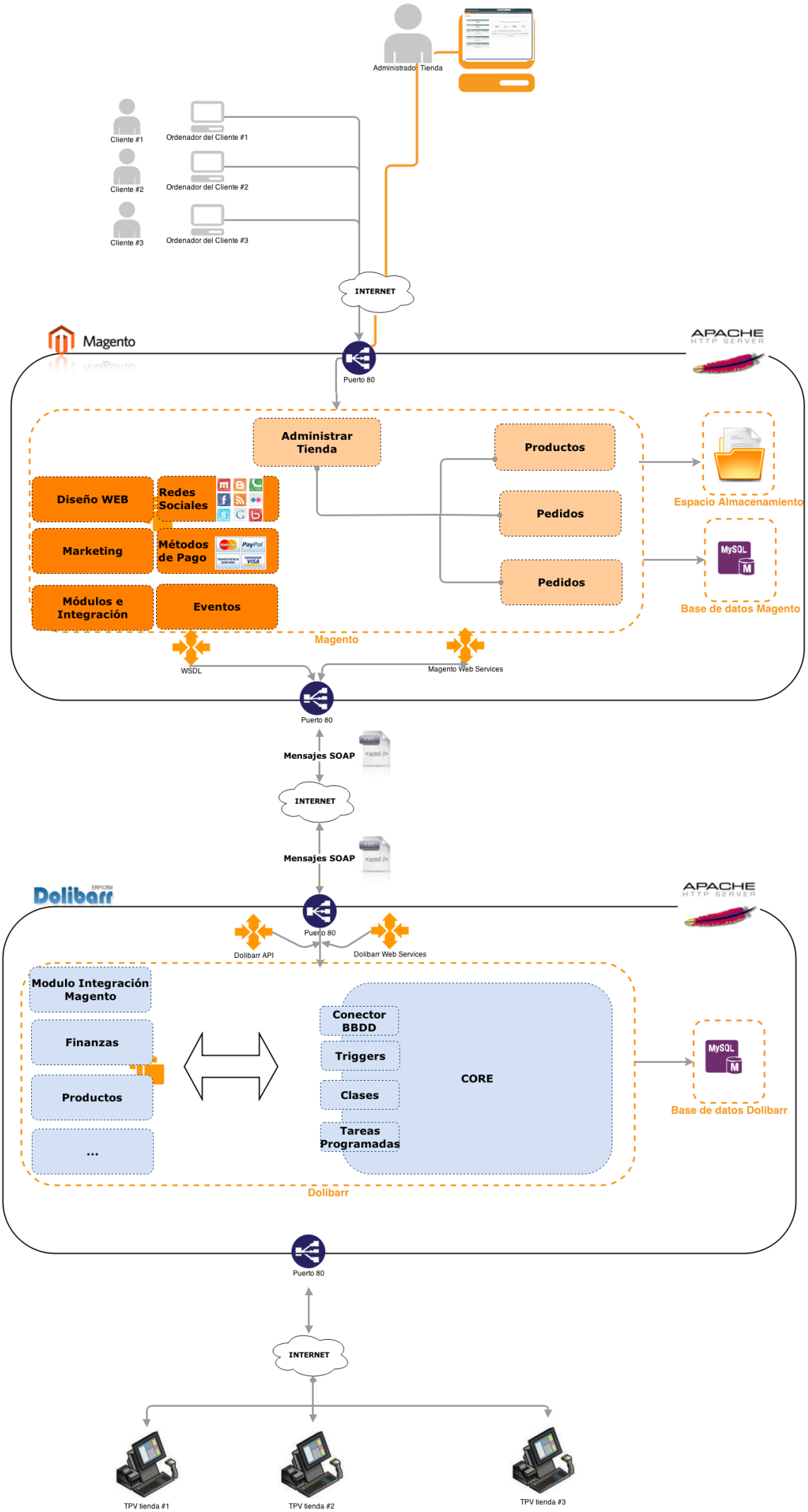


Ilustración 24: Arquitectura general

7.4. Implementación

Una vez terminada la fase de diseño, comienzan las tareas de instalación y desarrollo. Existen dos fases en la implementación. Por un lado la parte de configurar servidores e instalación de software (Apache, Magento y Dolibarr), y por otra parte el desarrollo de los módulos específicos.

En primer lugar, se requiere un servidor donde almacenar las aplicaciones, por lo que se contrata un servidor *Web* en la empresa digitalocean.com. Se trata de un servidor Ubuntu 12.04, en el cual es necesario configurar la arquitectura LAMP (Linux, Apache, MySQL y PHP). Los requisitos del servidor se han obtenido de los requisitos de Magento, en el apartado de requisitos en su *Web* [17]. Para ver paso a paso la configuración del servidor contratado, se puede consultar el Anexo 6: Configuración servidor LAMP.

La instalación y posterior configuración de Magento se detalla en el Anexo 7: Instalación Magento y la instalación del ERP Dolibarr en el Anexo 8: Instalación Dolibarr.

Teniendo ya la información suficiente sobre la solución tecnológica que se implementará en la PYME y creada la infraestructura en el servidor, se comienza una búsqueda en las tiendas de extensiones de Magento y Dolibarr, con el objetivo de buscar módulos de integración. Tras realizar una intensa búsqueda, solo se ha encontrado un módulo que integraba datos entre Magento y Dolibarr, llamado AguriaDolibarr API. Este módulo no servía para llevar a cabo el caso de estudio planteado por las razones siguientes:

- La última actualización del módulo es de la versión 1.7 de Magento, lo que hace que no funcione correctamente en la actual versión de Magento 1.9.
- La información que puede llegar a sincronizarse era muy limitada.

Por lo tanto, al no existir *software* que satisfaga las necesidades del caso de estudio, se decide llevar a cabo la implementación de una nueva solución.

La implementación de los módulos de Dolibarr y Magento, va a crear una comunicación asíncrona entre aplicaciones, con el objetivo de tener sincronizados los elementos necesarios para satisfacer los casos de uso, estos son: nombre, referencia, stock (inventario), descripción y otros atributos secundarios como peso, volumen etc.

En la tabla siguiente, se muestra un mapeo entre los distintos datos que se sincronizarán entre las BBDD de Dolibarr y Magento.

	Tabla BBDD Dolibarr	Columna Dolibarr	Tabla BBDD Magento	Columna Magento
Nombre	llx_product_stock	label	catalog_product_entity	weight
Referencia	llx_product	ref	catalog_product_entity	sku
Stock	llx_product_stock	reel	cataloginventory_stock_item	qty
Precio	llx_product	price	catalog_product_index_price	price
Descripción	llx_product	description	catalog_product_entity_text	value
Peso	llx_product	weight	catalog_product_flat_1	weight

Tabla 34: Mapeo de datos entre Dolibarr y Magento

La descripción detallada sobre el desarrollo de la extensión para Dolibarr se puede encontrar en el apartado 8.1, y la descripción del desarrollo de la extensión en Magento se encuentra en el apartado 8.2. Estas implementaciones tienen el objetivo de satisfacer todos los requisitos definidos.

7.5. Evaluación y pruebas

Tras la implantación de las distintas aplicaciones y el desarrollo de las nuevas funcionalidades, se pasa al diseño, ejecución y evaluación de las pruebas del sistema. Esto se detalla en el Capítulo 9 Evaluación y pruebas. Con desarrollo de esta fase concluye la etapa de implantación del sistema, por lo que se pasa a describir los trabajos futuros.

7.6. Mantenimiento

Siguiendo la metodología, la siguiente fase sería el mantenimiento futuro de la *Web* de comercio electrónico y el ERP. El mantenimiento común para ambas plataformas conlleva las siguientes tareas:

- Actualización a las nuevas versiones de cada plataforma, ya que cada pocos meses es necesario actualizar a las nuevas versiones del *software.* De esta manera se corrigen fallos de seguridad a la vez que se mejoran o añaden funcionalidades.
- Corrección de los errores en la solución inicial.
- Adaptación a nuevos requisitos legales, como es el caso de la ley de cookies, ley de privacidad, etc.
- Solucionar dudas sobre aspectos de configuración o manejo de la aplicación.
- Sistema de copias de seguridad, que garantice el almacenamiento de la información de la compañía.

Puntualizando más en el mantenimiento de Magento, y aunque no sea el objetivo de este proyecto, se recomienda elaborar un plan de marketing, ya que esta será la tarea que más recursos necesite una vez el proyecto esté operativo.

Para finalizar, se muestra el aspecto final de las dos plataformas. Estas se encuentran operativas y se pueden encontrar en los siguientes enlaces:

Magento: http://95.85.21.243

Magento-panel de control: http://95.85.21.243/index.php/admin
- usuario: *invitado*
- contraseña: *uc3minvitado*

Dolibarr: http://95.85.21.243/dolibarr/
- usuario: *invitado*
- contraseña: *uc3minvitado*

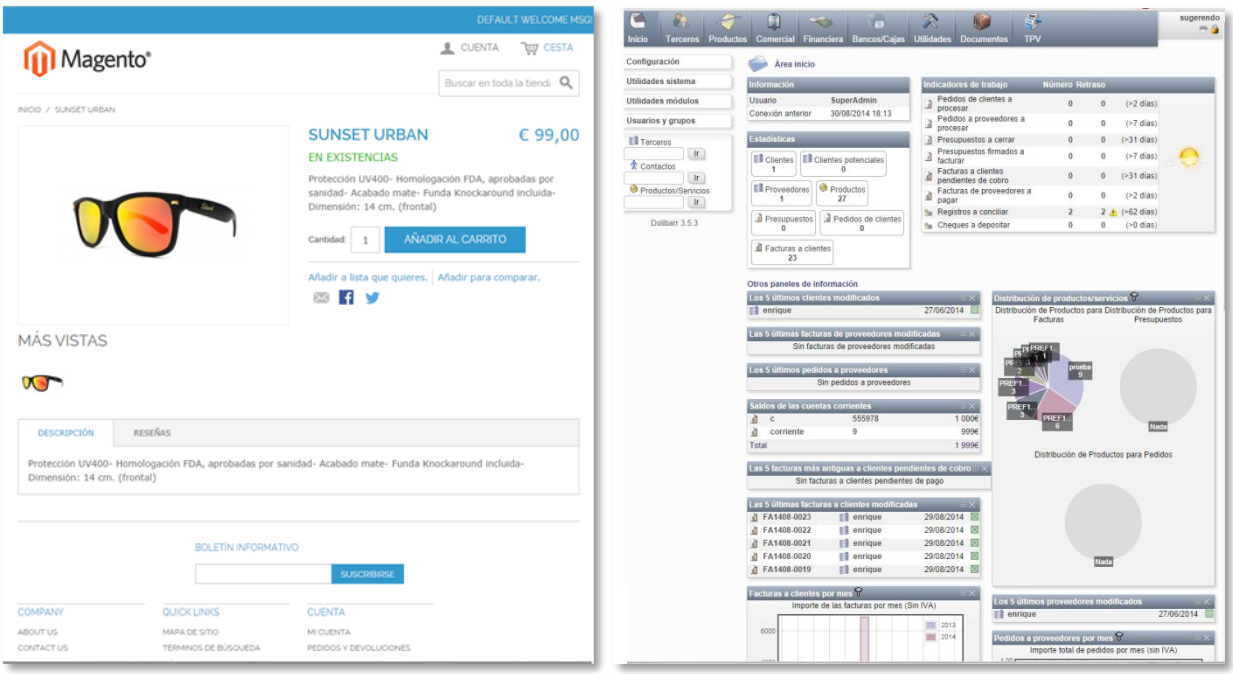


Ilustración 25: Portada final Magento y Dolibarr

Capítulo 8. Software desarrollado

En este capítulo se detalla el desarrollo de los módulos de la solución diseñada. Se especifica en detalle los pasos a seguir para conseguir que las extensiones estén operativas y preparadas para realizar las funcionalidades para la integración, dentro de sus respectivos sistemas.

8.1. Módulo Dolibarr

Dolibarr se compone de decenas de módulos en su instalación básica. Crear un módulo en Dolibarr requiere conocimientos de programación, en concreto de PHP. Por lo que para la creación de un nuevo módulo es necesario seguir una serie de directrices [18]:

En primer lugar, es necesario crear una estructura de ficheros como la mostrada en la ilustración inferior. Al nuevo módulo se le conoce como "moduloMagento", ya que su función es comunicarse con Magento.

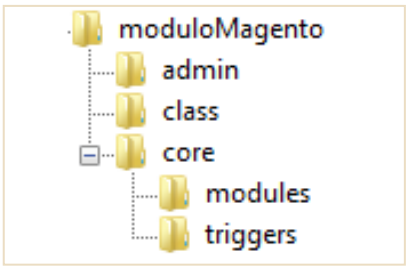


Ilustración 26: Estructura Módulo Dolibarr

En segundo lugar, es necesario crear un fichero descriptor del módulo. El nombre ha de ser el siguiente: *modmoduloMagento.class.php,* este fichero se basa en el esqueleto: *modMyModule.class.php,* que es un documento de ejemplo que sirve para configurar un módulo y sus elementos. En este fichero se detallarán las clases que componen el módulo, si tiene eventos, si tiene interfaz de administrador, o si necesita pestañas en el panel de usuarios.

Tras rellenar el fichero se puede comprobar que se ha configurado correctamente el módulo, accediendo en Dolibarr al *apartado del menú,* configuración → módulos*:*

Ilustración 27: Menú de Módulos instalados en Dolibarr

Como se quiere que el módulo responda a ciertos eventos, es necesario hacer uso de los *triggers* (disparadores) de Dolibarr. Los *triggers* son un mecanismo para realizar determinadas acciones cuando Dolibarr ejecuta y procesa una acción asíncrona (creación de factura, edición de un usuario, eliminación de un pedido, etc.) [19]. Para añadir un nuevo *trigger* a la extensión que se está desarrollando, es necesario editar el fichero creado en el apartado anterior (*modmoduloMagento.class.php)* con la siguiente línea:

```
$this->module_parts = array('triggers' => 1);
```

Ilustración 28: Código para añadir *triggers* en Dolibarr

Con lo detallado en el apartado anterior, Dolibarr ya sabe que este módulo va a escuchar eventos. Pero aun no sabe en concreto ante cual actuar.

En tercer lugar, se establecen los eventos que se necesita procesar. Para escuchar un evento es necesario crear un archivo con el nombre exacto: *interface_XX_all_Abc.class.php, d*onde XX es un número único entre 00 y 99. Y donde Abc son unas letras que identifiquen el archivo, en este caso se crea un nuevo fichero llamado: *interface_99_all_Pro.class.ph.* La nomenclatura de los ficheros es esencial para el correcto funcionamiento, por lo que es importante escribir el nombre del fichero correctamente.

Para escuchar el evento que se necesita, hay que establecer un condicional (*if*). Si esta se cumple, se pasa a ejecutar el código "dentro" del evento.

```
    function run_trigger($action,$object,$user,$langs,$conf){
         // Data and type of action are stored into $object and $action
        if ($action == 'PRODUCT_CREATE'){
              try{

        create_product($object->ref, $object->libelle, $object->price,0,
        $object->description);
                    }
                    catch(Exception $e){
        dol_syslog("problemas creando el archivo con sku: ".$object->ref);
                    }


         }
        elseif ($action == 'PRODUCT_PRICE_MODIFY'){ .... }

}
```

Ilustración 29: Código Dolibarr para eventos.

Como se ve en el código superior, en el módulo que se ha desarrollando, cuando se dispara el evento "PRODUCT_CREATE", se llama a una función creada específicamente para comunicarse con Magento. A esta función se le pasan unos parámetros, que son los recogidos en el formulario del ERP. Estos parámetros se obtienen del objeto "$object", que es creado por el sistema al provocar un evento. Durante el desarrollo, se observa que los atributos del objeto no se encuentran bien definidos en la documentación de Dolibarr, por lo que hay que investigar en el código "*core*" de Dolibarr, la manera de conocer el nombre de los atributos. Por ejemplo, para obtener los nombres de los atributos del código superior fue necesario buscar la clase producto y buscar qué atributos existían. Después, razonar cuál era el que se necesitaba en cada caso, ya que por ejemplo, la clase producto tiene un atributo para definir el stock máximo permitido, el stock actual en el sistema, el stock deseado y el stock vendido por el TPV. Esto provocó algunos retardos en el desarrollo.

Una vez explicada la forma en la que se procesa un evento en Dolibarr, se pasa a detallar la manera en la que se crea una comunicación vía Servicios Web utilizando SOAP. En este caso, se comenzará a describir la creación de un producto en la base de datos de Magento desde el Módulo de Dolibarr.

Para esta implementación, el código del evento deriva en la función "create_product", que tras recibir los parámetros de un evento u otro módulo, crea un nuevo cliente (proxy) de SOAP en base al WSDL de la tienda *online*.

Con el cliente ya creado, es necesario introducir un usuario y contraseña para poder invocar a los métodos de la tienda *online* desde Dolibarr. Se requiere una autenticación para poder acceder a los métodos que ofrece la interfaz ofrecida por el Servicio Web. Esta autenticación (método *login)* se configura en la consola de Magento en: Sistem→ Servicios Web →SOAP/XML-RPC→ users (ilustración inferior).

Ilustración 30: Creación usuario Servicio Web SOAP en Magento

Cuando el registro es satisfactorio, ya se tiene una *id* para la sesión. Con ella ya se pueden invocar métodos de Magento desde Dolibarr. En el ejemplo del código de la Ilustración 31 el cliente creado llama al método *catalogProductCreate,* el cual crea un producto con los parámetros que se le pasan. Si todo va bien, tras unas décimas de segundo se tendrá el producto creado en las BBDD de Dolibarr (ya que estamos en el módulo de Dolibarr) y Magento (se crea mediante un Servicio Web).

```
function create_product($sku,$name,$price,$qty,$description){


$proxy = new SoapClient('http://95.85.21.243/index.php/api/v2_soap/?wsdl');
$sessionId = $proxy->login('usr_doli', ******);

      $attributeSets = $proxy->catalogProductAttributeSetList($sessionId);
      $attributeSet = current($attributeSets);
      $stockItemData = array(
             'qty' => $qty, // cantidad
             'is_in_stock ' => 1, //activarlo
             ...
             );

      $result_id    =    $proxy->catalogProductCreate($sessionId,    'simple',
      $attributeSet->set_id, $sku, array(
             'name' => $name,
             'description' => $description,
             'url_key' => $name."-".$sku,
             'visibility' => '4',
             'price' => $price,
             'stock_data' => $stockItemData
             ...
             ));
   ...
  }
```

Ilustración 31: Función create_product en Dolibarr.

Si la comunicación es correcta, se recibe otro mensaje SOAP del servidor, con un valor positivo (Ilustración 48: Respuesta a un mensaje SOAP de Dolibarr). Y un mensaje con valor negativo, si la comunicación no se desarrolló correctamente

Los datos que se sincronizan en el caso de crear producto son: nombre, descripción, referencia, precio, nivel de stock y peso.

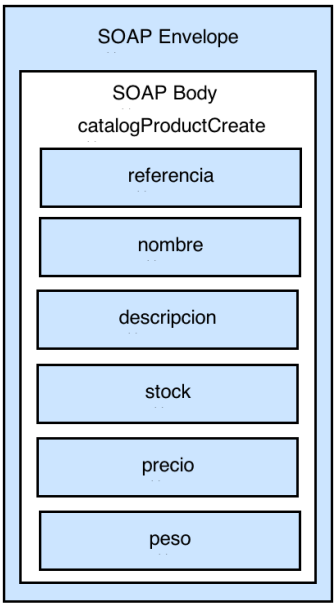


Ilustración 32: Mensaje SOAP para crear un producto

La creación del evento y la llamada de un Servicio Web del ejemplo anterior solo muestra uno de los eventos y llamadas que se han implementado en este Módulo. No obstante, implementar cada caso de uso ha tenido una dificultad distinta, ya que todos los escenarios requieren una información distinta, o el uso de ciertos métodos. Todas las funcionalidades del apartado casos de uso han sido implementados. Para entender la magnitud del módulo implementado, se listan los eventos que también se han utilizado en Dolibarr, con el correspondiente mensaje SOAP:

-PRODUCT_DELETE: Eliminación producto.

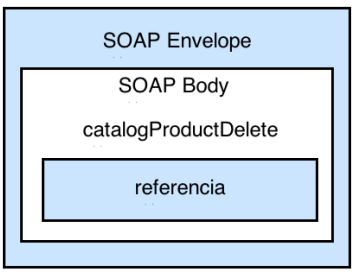


Ilustración 33: Mensaje SOAP para borrar un producto.

-PRODUCT_PRICE_MODIFY: cambio de valor en el precio.

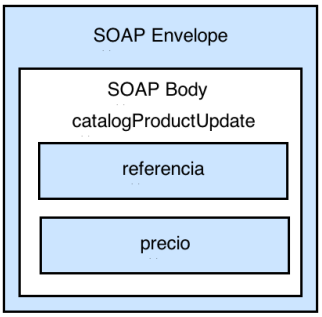


Ilustración 34: Mensaje SOAP para cambiar el precio de un producto.

-PRODUCT_MODIFY: modificación general de un producto.

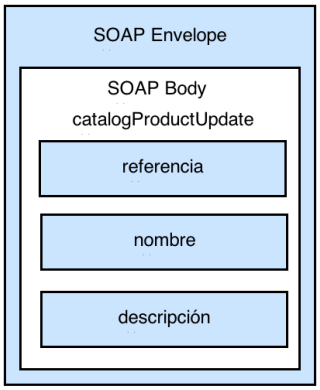


Ilustración 35: Mensaje SOAP para editar un producto.

-STOCK_MOVEMENT: modificación del stock de un producto.

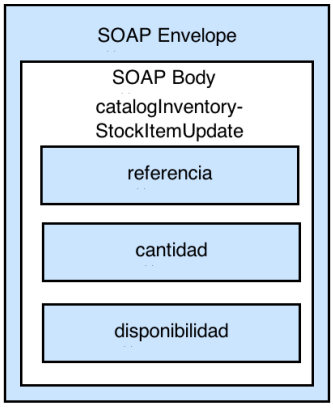


Ilustración 36: Mensaje SOAP para notificar un movimiento de inventario.

-LINEBILL_INSERT (producto vendido en el TPV)

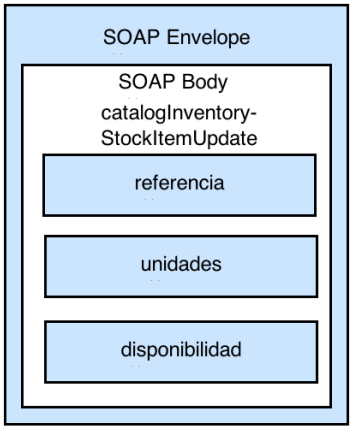


Ilustración 37: Mensaje SOAP para cambiar stock tras una venta.

Estos mensajes siempre reciben una respuesta como la que se puede ver en ilustración siguiente, donde "XXXX" es el nombre del método que envía el mensaje.

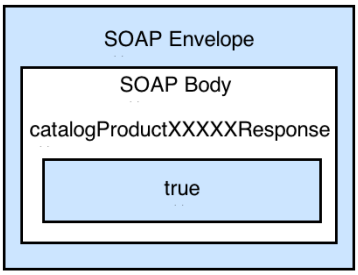


Ilustración 38: Mensaje SOAP de respuesta.

Una vez finalizada la integración asíncrona, para aportar más robustez al sistema, se ha decidido complementar el módulo con un script de ejecución periódica (*Cron*).

Este *script* se encarga de verificar que los productos que existen en ambas plataformas están totalmente sincronizados. Ha sido necesaria la implementación de este *script*, ya que si por alguna razón una petición de cambio de inventario no se realizase de forma correcta, este pequeño *script* podría corregir el inventario.

Además sirve para cargar productos desde Dolibarr hacia Magento en los siguientes casos:

- Cuando se importan productos, por primera vez, de forma masiva desde un fichero (.csv o .xml) a Dolibarr. Esta acción es muy útil, ya que no se producen eventos en el sistema de Dolibarr cuando se realiza una importación desde un elemento externo. Por lo que es necesario utilizar esta herramienta para así tener los productos sincronizados.

- También es necesario este *script, s*i se tiene Dolibarr en funcionamiento, y a continuación se instala el actual módulo desarrollado de integración.

En Dolibarr ya estarían todos los productos creados, entonces el evento "crear producto" no va a saltar para los productos que ya existían, por lo que es necesario forzar el *script* actual.

A continuación, se muestra el diagrama de flujo de la herramienta de sincronización periódica.

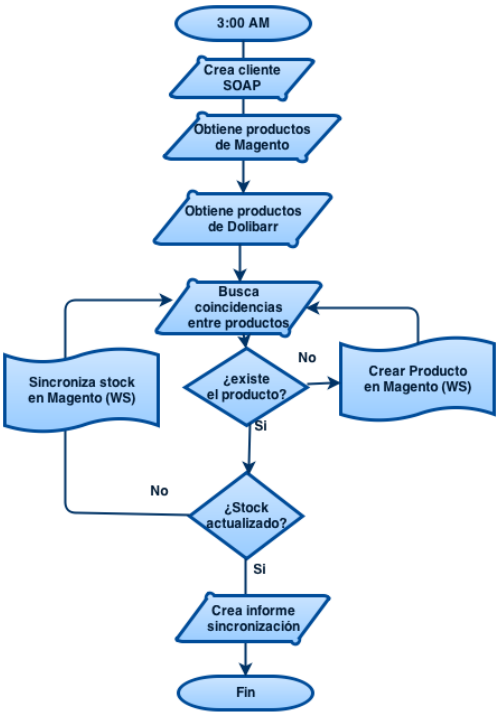


Ilustración 39: Diagrama de flujo sincronización.

Se detectó que había un problema cada vez que los productos se creaban en Magento desde Dolibarr mediante Servicios Web, ya que no se activaba el campo "Gestionar Existencias".

Para solucionar ese problema, hubo que añadir el atributo 'manage_stock' al *array* de stock de producto en el código fuente del módulo, y que así este atributo también viajaría vía Servicios Web y entonces crearía correctamente en Magento el nuevo producto.

```
'manage_stock ' => 0,
'use_config_manage_stock' => 1,
```

Ilustración 40: Código crear producto Magento desde Dolibarr

Para comprobar que se ha solucionado el problema anterior, se puede ver cómo antes *'manage_stock'* (gestionar existencias), tenía valor 1 y *'use_config_manage_stock'* tenía valor 0. Según se ve en la ilustración superior las existencias se gestionan como se establecen en los requisitos. Quedando la configuración en el panel de administración de Magento:

Ilustración 41: Gestionar existencias, panel Magento

Con la corrección del error anterior, se da por finalizada la implementación del módulo que conecta Dolibarr con la API de Magento.

8.2. Módulo Magento

Una vez finalizado el módulo que comunica Dolibarr con Magento, se detalla el desarrollo del módulo que comunica Magento con Dolibarr [20].

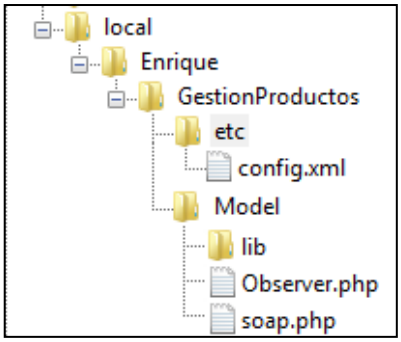


Tabla 35: Estructura módulo Magento

Al igual que pasaba en Dolibarr, hay que crear un fichero descriptor del módulo. En el caso de Magento, es el fichero *config.xml. E*ste fichero es la pieza clave para que el núcleo de Magento reconozca, que en este directorio existe una nueva extensión.

Como se puede contemplar en la ilustración inferior, este archivo contiene unas etiquetas XML que definen el módulo con una nomenclatura muy rígida [21]. Para definir una nueva extensión, primero se crea un elemento que contenga el nombre de la entidad que crea el módulo, en este caso "Enrique" seguido del título exacto del módulo (coincide con el directorio de la ilustración anterior). Además, en este fichero XML, se identifican dentro de la etiqueta <events> los eventos a los que este módulo de Magento se ha de subscribir. Para la realización de la extensión "GestionProductos", se ha subscrito al evento *checkoutSubmitAllAfter,* que es un evento que se produce al realizar una compra en la tienda *online* de forma satisfactoria.

```
<config>
     <modules>
        <Enrique_GestionProductos>
            <version>1.0.1</version>
        </Enrique_GestionProductos>
    </modules>
```

```
        <frontend>
        <events>
           <checkout_submit_all_after>
               <observers>
                  <Enrique_GestionProductos_Model_Observer>
                         <type>singleton</type>
                           <class>Enrique_GestionProductos_Model_Observer</class>
                         <method>checkoutSubmitAllAfter</method>
                  </Enrique_GestionProductos_Model_Observer>
                  </observers>
           </checkout_submit_all_after>
        </events>
 </frontend>
</config>
```

Ilustración 42: Fichero config.xml

Llegado a este punto, se tiene un módulo creado con su estructura de directorios y ficheros, pero no existe aún nada de la lógica del programa. Entonces para poder empezar a escribir código PHP, se ha de crear un fichero *Observer.php* dentro de la carpeta *Model,* que pueda escuchar eventos de Magento. En la siguiente clase PHP se implementan los casos de uso que se desarrollan en la tienda *online*, es decir, cuando se crea un nuevo pedido salta un nuevo evento, dentro de este se ha de recopilar toda la información del pedido; los productos vendidos y su cantidad, e informar a Dolibarr mediante Servicios Web.

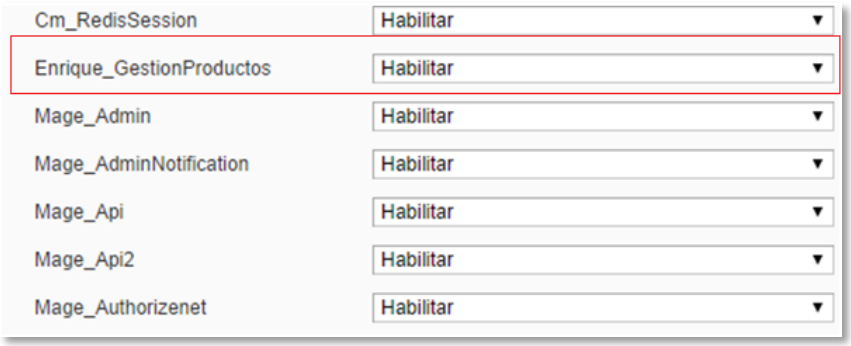


Ilustración 43: Módulos instalados en Magento

Para la implementación de un cliente SOAP, se hace uso de la librería NuSOAP. El cliente para los Servicios Web se obtiene del WSDL de Dolibarr. Las acciones que se realizan con el cliente SOAP son:

-Autenticarse: se introducen los datos de usuario de Dolibarr, además de la clave para usar los Servicios Web, en este caso la clave de seguridad es una cadena alfanumérica de 32 caracteres.

-Hacer una llamada al Servicio Web ' *updateStockProduct* ' por cada producto que se ha comprado, con los parámetros: *'authentication'*, *'productRef'* y *'qty'*, que son el *token* de autenticación, el número de referencia del

producto que se modifica desde Magento en Dolibarr, y la cantidad del producto.

```
<?php
class Enrique_GestionProductos_Model_Observer extends Mage_Core_Model_Abstract
{
   public function checkoutSubmitAllAfter(Varien_Event_Observer $observer)
   {
       require_once('lib/nusoap.php');         // Include NuSOAP

       $WS_DOL_URL                                                              =
'http://95.85.21.243/dolibarr/webservices/server_productorservice.php';
       $WS_METHOD = 'updateStockProduct';
       // Set the WebService URL
       $soapclient = new nusoap_client($WS_DOL_URL);

       if ($soapclient)
       {
              $soapclient->soap_defencoding='UTF-8';
       }
       // Call the WebService method and store its result in $result.
       $authentication=array(
              'dolibarrkey'=>'**********************************',
              'sourceapplication'=>'DEMO',
              'login'=>'sugerendo',
              'password'=>'sugerendo',
              'entity'=>'');


        Mage::log('Evento fin de compra !!!' , null , 'system.log');
              $order = $observer->getEvent()->getData('order');
              $array_items=$order->getAllItems();
              for($i=0;$i<count ($array_items); $i++){
                     $item=$array_items[$i];
                     $sku=$item->getProduct()->getSku();
                     $qty=$item->getQtyOrdered();
                     $parameters                                                =
array('authentication'=>$authentication,'productRef'=>$sku,'qty'=>$qty);
                     $result = $soapclient->call($WS_METHOD,$parameters);

                                          }
        Mage::log('Evento fin de compra!' , null , 'system.log');
         }
}
?>
```

Ilustración 44: Clase PHP del módulo Magento

El mensaje de tipo SOAP que se enviaría hacia Dolibarr cada vez que se produzca una venta sería:

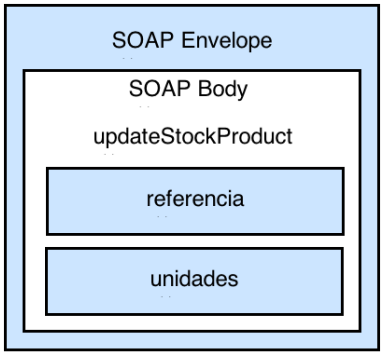


Ilustración 45: Mensaje SOAP notificación venta.

Con la finalización del desarrollo del módulo para Magento, se concluye la etapa de implantación. La siguiente etapa consistirá en realizar una evaluación al sistema, para poder comprobar su correcto funcionamiento.

Capítulo 9. Evaluación y pruebas.

En este capítulo se detallan las pruebas realizadas en el sistema, una vez se encuentra finalizada la implementación. Se van a realizar cuatro tipos de verificaciones.

- En primer lugar, se han capturado los mensajes de tipo SOAP que se intercambian entre Magento y Dolibarr.
- En segundo lugar, se han realizado pruebas unitarias con SOAPUI. Esta herramienta de código abierto permite realizar baterías de pruebas a los distintos Servicios Web utilizados.
- En tercer lugar, se han realizado pruebas de caja negra con todas las entradas posibles del sistema.
- Por último, se realizan pruebas de rendimiento, para comprobar la capacidad del servidor.

Se ha comenzado capturando con el analizador de paquetes Wireshark, los mensajes SOAP que se envían desde Dolibarr a Magento. Se pretende evaluar si se envían correctamente a través de la red todos los campos. Además se pretende mejorar la compresión sobre los mensajes de tipo SOAP, los cuales se explican en el apartado 3.3.1: SOAP.

Tras filtrar los mensajes, se ve cómo se intercambian peticiones y respuestas HTTP/XML, las cuales contienen mensajes SOAP. En primer lugar se puede ver la petición y a continuación la respuesta:

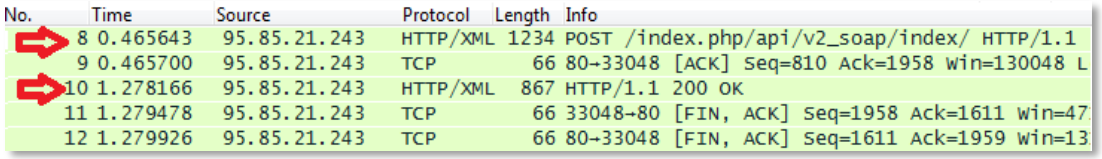


Ilustración 46: Trazas capturadas en Wireshark

Pasando a examinar el contenido de estas peticiones, se observa que el contenido es un fichero XML en el que se observa la estructura del mensaje SOAP,

donde se encuentran los distintos elementos que envía la aplicación más la ID de sesión.

En el mensaje inferior se observa:

- *"sesion ID"*: Token de sesión que identifica a la sesión del mensaje como una sesión ya registrada en el sistema.
- *"product"*: Referencia (SKU) del producto.
- *"productData"*: Es la entidad que representa al producto. En ella están los campos de: *name*, *description* y *weight*.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:Magento"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
         <ns1:catalogProductUpdate>
               <sessionId xsi:type="xsd:string">73b0c00e180696***</sessionId>
               <product xsi:type="xsd:string">REF123459</product>
               <productData xsi:type="ns1:catalogProductCreateEntity">
                     <name xsi:type="xsd:string">Isson – Felix BLACK</name>
                     <description xsi:type="xsd:string">Inspirados en la
                     arquitectura europea contemporánea y en el diseño
                     industrial, son garantía de resistencia y durabilidad.
                     </description>
                     <weight xsi:type="xsd:string">0.5</weight>
               </productData>
               <storeView xsi:nil="true"/>
               <identifierType xsi:nil="true"/>
         </ns1:catalogProductUpdate>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ilustración 47: Mensaje real SOAP de Dolibarr

En el fichero XML inferior se observa la respuesta al mensaje anterior.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:Magento" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

   <SOAP-ENV:Body>
         <ns1:catalogProductUpdateResponse>
         <result xsi:type="xsd:boolean">true</result>
         </ns1:catalogProductUpdateResponse>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Ilustración 48: Respuesta a un mensaje SOAP de Dolibarr

Por otra parte, se tienen las **pruebas unitarias**. Estas se han desarrollado utilizando la herramienta SOAPUI, una aplicación versátil que permite testear, simular y generar código de servicios *Web* de forma rápida y sencilla.

Se ha creado una batería de pruebas para poder testear el sistema de manera rápida cada vez que se produzca un cambio en el código. En primer lugar, para realizar este tipo de pruebas, es necesario configurar los distintos métodos que se quieren testear, en este caso: *login*, *catalogProductUpdate*, *catalogProductCreate* y *catalogProductDelete*. A continuación, es necesario obtener el *token* de sesión que devuelve "login" para poder realizar las siguientes pruebas. Una vez se tiene un *token* con el que poder hacer nuevas llamadas al servidor, se configuran los campos requeridos para los otros tres métodos. Para poder ejecutar las pruebas una y otra vez, es necesario configurar los mensajes de la siguiente forma: se rellenan los campos de un nuevo producto con el método "*catalogProductCreate*", se envían. Se modifican los datos de dicho producto con el método "*catalogProductUpdate*", y se envían modificados. Por último se Borra el producto con el método "*catalogProductDelete*".

Se realizan las pruebas y todas dan un resultado positivo. Además se observa que el tiempo empleado en todas las pruebas cumple con el requisito RNF-004, que decía que el retardo debía ser menor de 5 segundos.

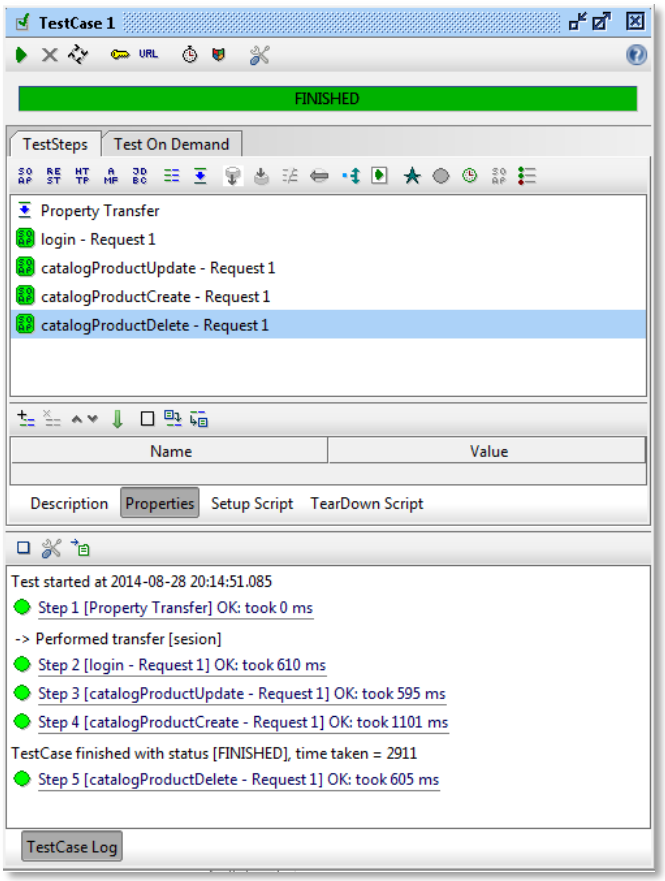


Ilustración 49: Resultado de las pruebas unitarias con SOAPUI

Para finalizar la fase de pruebas, se van a realizar varias pruebas de caja negra, que pretenden analizar la funcionalidad del sistema desde las vistas de Magento y Dolibarr. En cada aplicación se han realizado una serie de pruebas. En primer lugar se muestran las pruebas de **Dolibarr**.

- Pruebas para la creación de nuevos productos desde Dolibarr:

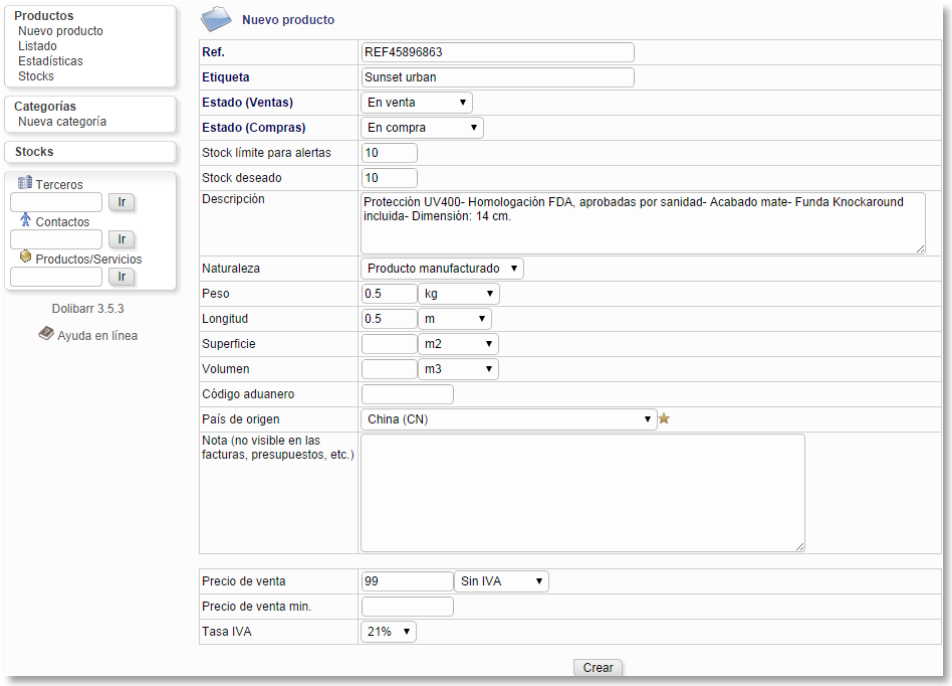


Ilustración 50: Pruebas del formulario de creación de productos en Dolibarr

En la imagen superior se introducen los datos obligatorios para la creación de un nuevo producto en Dolibarr. Estos son enviados a Magento correctamente como se ve en la imagen inferior.

Se comprueba que si se escribe un número de referencia ya existente en el sistema, Dolibarr produce un error, lo que es correcto.

También se comprueba que si un campo no obligatorio no es rellenado, este viaja vacío hacia Magento, por lo que también es correcto.

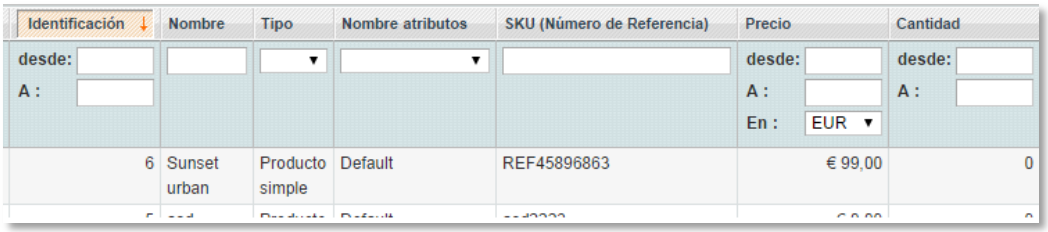


Ilustración 51: Producto creado en Magento

- Comprobación para modificar existencias manualmente desde Dolibarr.

Ahora se van a realizar pruebas de movimiento de stock. El formulario de creación de un producto, no permite introducir los niveles de stock en el almacén. Además cada vez que la empresa reciba nuevos productos para poner a la venta ha de introducir la cantidad en el ERP, por lo que se van a realizar pruebas con el fin de determinar si se realizan correctamente los movimientos de stock.

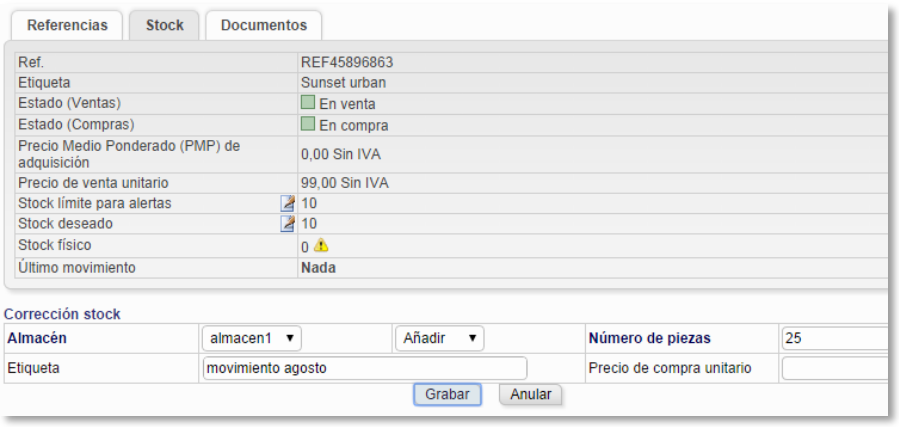


Ilustración 52: Pruebas formulario modificación stock Dolibarr

Se han realizado pruebas para:

- Determinar la cantidad máxima de stock, en Magento la cantidad máxima es: 99999999.9999 unidades, mientras que en Dolibarr es $9.0 \cdot 10^{65}$ unidades.
- Comprobar si se pueden alcanzar valores negativos.
- Comprobar si se pueden introducir caracteres distintos de números.
- Comprobar que todo el stock siempre permanece sincronizado.

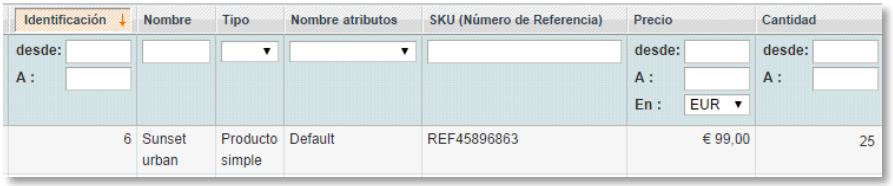


Ilustración 53: Stock actualizado Magento

Las pruebas han finalizado con éxito.

- Comprobación para crear pedido en el TPV.

Las siguientes pruebas diseñadas tienen como objetivo evaluar el correcto funcionamiento de distintas entradas en el TPV de las tiendas físicas.

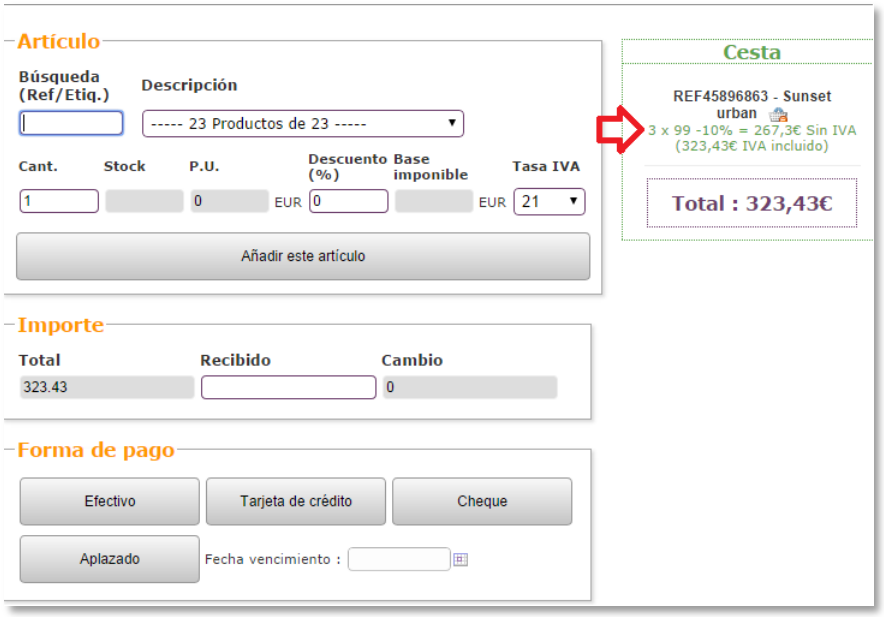


Ilustración 54: Pruebas TPV tiendas físicas

Se han realizado pruebas para:

- Determinar que tras vender satisfactoriamente X unidades, estas son restadas también en la BBDD de Magento. En la ilustración superior se ve cómo se produce la venta de 3 unidades, y estas se descuentan en Magento correctamente, como se ve en la imagen inferior.
- Determinar que si se quiere vender más unidades de las existentes en stock, no se finaliza correctamente la compra.

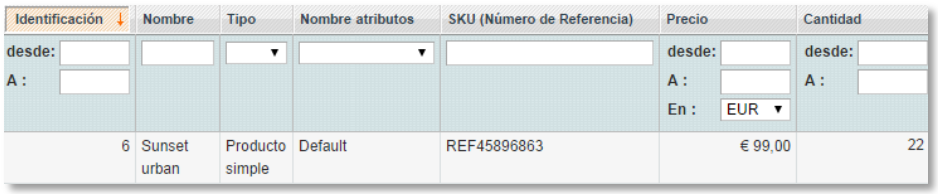


Ilustración 55: Stock en Magento tras pedido TPV

Las pruebas han finalizado con éxito.

- Pruebas para modificar un producto en Dolibarr.

Se ha comprobado, que se pueden modificar correctamente los siguientes cambios: nombre del producto, descripción del producto, peso y precio.

Además se verifica que la referencia no puede ser modificada, ya que es el elemento fijo en el que se realiza la sincronización, como se enumera en el requisito: RNF-003.

- Pruebas para comprobar la eliminación de un producto en Magento y Dolibarr.

Se ha verificado la acción de eliminar producto desde el ERP, la cual se propaga hacia Magento.

En segundo lugar, se muestran las pruebas con **Magento**.

- Prueba de nuevo pedido en la tienda *online*.

El siguiente caso de uso a comprobar es el de realizar un pedido en la tienda *online*. Se va a realizar un proceso de compra completo.

Ilustración 56: Página producto Magento

- Para realizar la validación de la funcionalidad se selecciona un producto en existencias, y se añade una unidad al carro de compra.

- Se introducen los datos de envío y se paga el pedido con la tarjeta de la ilustración, que se ha creado de ejemplo para Magento, y se procede a finalizar el pago .

- Se comprueba que el stock ha descendido en una unidad, tanto en Magento como en Dolibarr, por lo que la comunicación entre estas aplicaciones también es correcta.

Ilustración 57: Stock en Dolibarr tras pedido en tienda online

Se ha medido la velocidad en la finalización del envío, que es la parte en la que se envía el mensaje SOAP hacia Dolibarr. Esta es de media 7.2 segundos, según diez pruebas realizadas.

Muestra	Tiempo
1	8.62
2	5.67
3	10.07
4	8.06
5	7.52
6	7.21
7	5.93
8	6.3
9	7.59
10	5.86
Media:	7.28

Tabla 36: Tiempos de realización del proceso de venta

Por último, se han realizado pruebas de rendimiento. Se ha utilizando una herramienta[16] para someter a un servidor web a un elevado número de conexiones concurrentes. Se ha comprobado que con 20 usuarios realizando peticiones concurrentemente, el servidor no superaba el 25% de su capacidad ni en memoria ni en CPU.

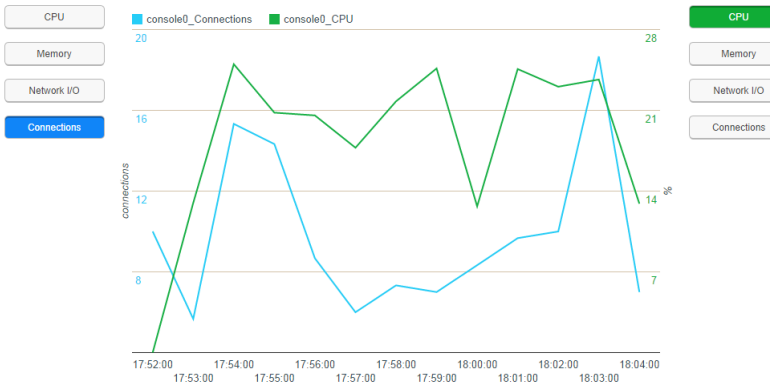


Ilustración 58: Pruebas de rendimiento del servidor

[16] Blazemeter: platform for back-end testing: http://blazemeter.com/features

Capítulo 10. Conclusions and future work

10.1. Conclusions

Once completed, the development of the current project will have achieved the goals initially set, as well as improved some aspects, for the proper functioning of a company in the sphere of electronic commerce, with the benefits of integration between different platforms.

One goal was to provide a methodology to deploy and integrate the technological infrastructure for SMEs. This methodology is aimed at companies that have an ecommerce platform and more than one shop. This methodology is expected to serve as a reference for other companies that develop their projects.

Personally, the project has been very productive for learning. I was able to learn and master tools I didn't know at the beginning of the project. I have also discovered what engineering integration is and how it can work between business applications. These achievements are in addition to learning a new programming language: PHP, without which there could have performed the development of modules of Dolibarr and Magento.

Working with open source applications has surprised me. I found that the free enterprise software can provide many benefits and save you lot of costs. In addition, having worked for the development community, creating new features that no one had created before has been very satisfying.

Definitely, the most difficult part for me in this project was the investigation and implementation of the technology platform. Exploring

different alternatives for ERPs and eCommerces open source as well as installing and understanding how to develop new features within them, was a great effort. Once the solution was well-designed, work began to roll and everything went quickly until now.

Regarding project management, I have to say that the initial estimation fell short; the size of the project has been slightly larger than estimated. To compensate, I had to work more hours in the phase of preparation of the report.

What really has me excited in this project has been the way I've learned autonomously. When I started in April, I had not mastered the topic yet, and step by step I got to the final solution with my work. I've learned that I can deal with any problem no matter what size it is.

10.2. Future Work

As to future work, there is no limit in an eCommerce. However, for this project it would be advisable to continue to perform more integration between data types between Dolibarr and Magento. It would be nice to integrate complete orders, users, customers, etc. But a project of integration between two business applications requires a development team for several months.

The implementation of CRM software would provide a benefit to improve customer service and sales management. This would be the next piece to be added to the topology.

One aspect to improve and was not in the scope of this project, is the design of the online store. Magento has been set for sale, but visually the work of a Web designer is necessary to make the visual appearance better.

When web traffic increases on the online store, it will be necessary to redesign the architecture of Magento. I will probably have to optimize the server, even in the case of large concurrent user growth, load balancing may be necessary.

Capítulo 11. Presupuesto

Este capítulo pretende realizar un cálculo detallado del proyecto, donde se desglosan los costes que implica realizar un proyecto de investigación e implantación de software empresarial.

Para la realización del cálculo, se debe tener en cuenta el tiempo empleado en cada tarea, los costos de personal, y el coste del material tanto *hardware* como *software*.

- **Coste Software**

Nombre	Coste	Dedicación (meses)	Periodo de depreciación(meses)	Coste
Microsoft Office Professional 2007	469,00€	6	30	93,8
Adobe Acrobat XI Pro	24,59€/mes	6	--	147,54 €
Dropbox	---	---	---	---
Magento	---	---	---	---
Dolibarr	---	---	---	---
MYSQL	---	---	---	---
Apache	---	---	---	---
Notepad++	---	---	---	---
PhpStorm	26,00 €	6	12	13,00 €
Magicento	20,00€	6	24	5,00 €
SoapUI	---	---	---	
Wireshark	---	---	---	---
Filezilla	---	---	---	---
Gantt Proyect	---	---	---	---
Navegador Chrome	---	---	---	---
			Total	259,34 €

Tabla 37: Costes de herramientas software.

- **Coste de los equipos empleados**

Descripción	Coste	Dedicación (meses)	Periodo de depreciación (meses)	Coste
Ordenador Sobremesa	800€	4	60	53,33 €
Ordenador portátil: HP Probook 4510s	529€	2	60	17,63 €
Servidor DigitalOcean	7,40€/mes	4	--	29,60 €
			Total:	100,56 €

Tabla 38: Costes de materiales hardware.

- **Coste del personal**

Este proyecto final de grado, realizado por Enrique Guadalupe Estévez como alumno, teniendo como director de proyecto a D. José Antonio Díaz Nicolás de la empresa Sugerendo Sistemas SL y como tutor de proyecto a D. Jesús Arias Fisteus profesor de la Universidad Carlos III de Madrid.

Nombre	Categoría	Dedicación (horas)	Coste (hora, sin IVA)	Coste
Enrique Guadalupe Estévez	Estudiante	400	25	10000,00 €
José Antonio Díaz Nicolás	Director de proyecto	60	35	2100,00 €
Jesús Arias Fisteus	Tutor de proyecto	30	35	1050,00 €
			Total:	13.150,00 €

Tabla 39: Costes de personal.

Total coste del proyecto sin costes indirectos:

Descripción	Coste
Software	259,34 €
Equipos	100,56 €
Personal	13.150,00 €
Total sin costes indirectos	13.509,90 €

Tabla 40: Costes totales antes de costes indirectos.

Además, es necesario citar en el coste total del proyecto, la parte proporcional debida a los gastos indirectos. Estos pertenecen a factores muy diversos tales como tarifas eléctricas, Internet, desplazamientos etc. Se ha estimado que los costes indirectos suponen una tasa del 20% sobre el coste del personal.

Por último, sobre los costes totales, es necesario añadir el IVA del 21%, para así obtener el coste final del proyecto.

Descripción	**Coste**
Total sin costes indirectos	13.509,90 €
Costes indirectos (20%)	2.630,00 €
Subtotal sin impuestos:	16.139,9 €
IVA (21%)	3.389,379 €
Total	**19.529,279 €**

Tabla 41: Resumen final de costes del proyecto

Capítulo 12. Entorno socio-económico: Comercio electrónico.

El desarrollo de este proyecto se va a llevar a cabo en el contexto económico y social de España, por ello es importante detallar el estado en el que se encuentra el comercio electrónico en España en el año 2014.

El comercio electrónico en España ha continuado con su crecimiento en el último año. En términos absolutos el comercio electrónico movió más de 12.380 millones de euros según revela el Estudio sobre Comercio Electrónico elaborado por el observatorio nacional de las telecomunicaciones y de la sociedad de la información (ONTSI) en 2013 [22], lo que supone un incremento de un 13.5% respecto al año anterior. Este crecimiento es lógico debido al incremento de compradores internautas en 2013, ya que ha pasado de 13,2 millones en 2011 a más de 15 millones en 2012.

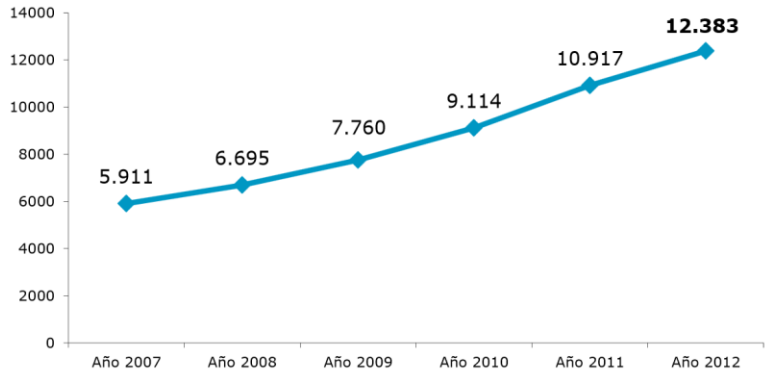


Ilustración 59: Volumen de comercio electrónico (Millones de Euros) [22]

El gasto medio por usuario a través de Internet en 2012 es de unos 815€ anuales, cifra que está disminuyendo levemente, ya que en 2011 era de 828€ anuales y en 2010 de 830€. Esta tendencia no es más que un síntoma de que el comercio electrónico también se ve afectado por el estado económico global de España. También esta bajada es debida a que antes, por el temor de ciertos usuarios por perder su dinero al pagar por Internet. Un solo cliente realizaba

pedidos para varios conocidos, siendo ahora mayor el número de usuarios pero que gastan menos de media.

Año	Importe total
2007	739 €
2008	754 €
2009	749 €
2010	831 €
2011	828 €
2012	816 €

Fuente: Panel Hogares, ONTSI

Ilustración 60: Evolución del gasto medio anual por comprador [22]

Por otro lado existe un gran crecimiento en el comercio electrónico desde dispositivos móviles, lo que se empieza a llamar m-commerce (comercio móvil). En 2012 el m-commerce era el 8% del total de las compras, unos 2,1 millones de usuarios, mientras que en 2013 ya superó el 10%, lo que hará que las empresas que quieran vender por Internet se tengan que involucrar en todos los canales posibles, lo que es conocido como multicanalidad.

Los productos que más se compran a través de Internet son los relacionados con el turismo y el ocio, ya que el 50% del total de las reservas del sector transportes se realizan por Internet, al igual que un 40% de las reservas de alojamiento, seguidas del 33% de compras electrónicas de entradas para espectáculos.

Las razones fundamentales por la que los usuarios compran por Internet son el precio (72%) y la comodidad (63%). Mientras que los lastres son: la curiosidad por no ver el producto que se está comprando, el no tratar con una persona físicamente y los gastos de envío.

Tendencia

Los consumidores cuando quieren comprar un producto, actualmente interactúan con empresas a través de distintos canales. Las empresas que no se sepan adaptar a los nuevos avances podrían encontrar dificultades para competir, e incluso quedarse atrasados del frenético camino que lleva el comercio actualmente. Hay diversos elementos que van a ser de vital importancia en el progreso empresarial: la analítica de grandes datos, la multicanalidad y el nuevo modelo de relaciones.

Se podría decir que la analítica de datos puede ser "la nueva materia prima" [23]. Se ha llegado a un punto en el que el "*Big Data*" y la tecnología de análisis está proporcionando información para resolver problemas hasta ahora irresolubles, tanto a empresas como a los gobiernos. Estos datos pueden servir tanto a las

empresas para desarrollar su plan de marketing, como a la policía para prevenir conductas delictivas.

Otro aspecto fundamental será la importancia de estar en todos los canales posibles. Las empresas lucharán por abarcar el mayor número de canales, ya que los consumidores están muy diversificados. Se puede ver cómo la tendencia del comercio electrónico va en aumento respecto al número de usuarios, pero también la venta en tienda sigue siendo la base de muchas empresas. El reto para este 2014 será conseguir que el canal del comercio móvil se popularice, ya que muchos usuarios están con su dispositivo móvil en la mano de camino al trabajo o a la universidad. Ese tiempo puede ser muy importante para que el usuario descubra los productos o realice las compras, y si no se está preparado para abarcar esos clientes se perderá un importante volumen de ventas. El comercio móvil puede alcanzar un volumen de ventas muy grande. Por ejemplo en el último *Black Friday* de Estados Unidos, se produjo casi un 22% de las ventas a través de tablets y teléfonos, unos 500 millones de dólares [24].

Respecto a las empresas, la tendencia está siendo la de apostar por la integración entre todos los recursos de los que dispone la empresa, conectando todos los puntos de venta para tener información a tiempo real de stock, ventas, usuarios, facturas y/o empleados.

Hoy en día para llevar a cabo la implantación de un comercio electrónico, es necesario utilizar una solución ya desarrollada. Realizar una plataforma de comercio electrónico desde cero sería muy caro, o probablemente poco profesional. Las plataformas de comercio electrónico se caracterizan principalmente por poder abarcar las necesidades básicas de una empresa con tan solo realizar la instalación del *software*.

Capítulo 13. Marco Regulador

Para comprobar la viabilidad legal del proyecto, se han analizado las leyes vigentes, comprobando que estas son aplicadas a la implantación de un proyecto de implantación e integración de comercio electrónico.

13.1. Real Decreto-ley 13/2012, de 30 de marzo- Cookies.

La primera ley a tener en cuenta, es la que establece, que las páginas web deben informar a los usuarios, sobre el almacenamiento de cookies en un servidor.

Una cookie es un fichero que envía el servidor donde se aloja la página web, al navegador del usuario. Este fichero almacena información sobre las preferencias, localización, idioma, etc. de un usuario.

La ley de cookies, Real Decreto-ley 13/2012, de 30 de marzo [25], obliga a los propietarios de páginas web a ofrecer al usuario del sitio web, la posibilidad de aceptar e instalar las cookies en sus ordenadores personales.

El proceso que se ha de ejecutar para cumplir la ley correctamente es:

- Informar al usuario de forma clara y completa, sobre las cookies que se van a instalar en su ordenador.
- Recibir el consentimiento por parte del usuario, para que se puedan instalar las cookies que se acaban de notificar.
- Instalar las cookies, después de la aceptación de establecimiento de estas.

13.2. Ley Orgánica de Protección de Datos (LOPD)

La siguiente ley en tener en cuenta, es la ley que permite proteger los datos privados de los usuarios. Los datos de usuarios que se utilizan, almacenen y con los que funciona un portal de comercio electrónico, están definidos como "datos

sensibles" según el artículo 7 de la LOPD [26], por lo que es obligatorio cumplir unas directrices para el cumplimiento de la LOPD. Debe tenerse en cuenta que:

- Cuando un usuario se registre en el sistema, previamente tiene que aceptar unos términos, en los cuales el usuario entiende y da permiso a la empresa responsable del comercio para almacenar y consultar sus datos de carácter personal.

- En el caso que el usuario introduzca información delicada en el sistema, y esta información sea almacenada por la empresa responsable del comercio. Si se produjese una pérdida de información, o algún tipo de filtración de datos, se responsabilizará directamente a la empresa que acumulaba estos datos, siendo posible que se le aplique una sanción según dicte la LOPD.

13.3. Ley General de Telecomunicaciones.

La siguiente ley aplicable establece unos criterios para establecer comunicaciones por medio de medios de telecomunicaciones.

En el caso de realizar un pago en un entorno cifrado, se tienen en cuenta según el artículo 43 de la Ley General de las Telecomunicaciones [27] que:

"*Cualquier tipo de información que se transmita por redes de comunicaciones electrónicas podrá ser protegida mediante procedimientos de cifrado*."

13.4. Ley General para la Defensa de los Consumidores y usuarios

Por último hay que tener en cuenta los derechos de los usuarios y clientes. Por lo que es obligatorio obedecer ciertos términos.

El comercializar productos desde una tienda *online*, conlleva adaptarse a la normativa establecida en la Ley General para la Defensa de los Consumidores y usuarios aprobada en el R.D.L. 1/2007, de 16 de noviembre. Esta normativa establece a día de hoy que:

- La información que esté a disposición del usuario para informarle sobre el precio, plazos de entrega y costes de devolución, debe estar planteada con antelación, en el caso que sea posible antes del proceso de compra.
- Existe un periodo de **catorce días** para devolver la compra si el consumidor no está conforme. Si este plazo no se encuentra correctamente indicado en el sitio web, el plazo pasará a ser **de un año y catorce días.**
- El plazo de entrega máximo es de **treinta días**.

- El proceso de devolución debe estar bien explicado, especialmente si se debe **abonar**, por parte de los usuarios, algún coste adicional para realizar una devolución.

Bibliografía

[1] D. S. Linthicum, «Enterprise Application Integration,» Addison-Wesley Professional, 2002, p. 400.

[2] A. Reeve, «Chapter 12. Data Integration Patterns,» de *Managing Data in Motion*, Morgan Kaufmann, 2013, p. 204.

[3] «Gartner,» IT Spending: How Do You Stack Up?, 2004. [En línea]. Available: http://www.gartner.com/research/attributes/attr_47450_115.pdf.

[4] J. Mula, «Material Requirement Planning with fuzzy constraints and fuzzy coefficients,» www.sciencedirect, 2006.

[5] D. Sinay, Web Services con C#, Argentina, 2006, p. 356.

[6] S. Weerawarana, F. Curbera y L. e. al, «Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More,» Prentice Hall, 2005, p. 456.

[7] F. Bolton, Pure CORBA, Sams, 2001.

[8] W. Grosso, Java RMI, O'Reilly Media, Inc., 2001.

[9] M. L. Bustamante, Learning WCF, O'Reilly Media, Inc., 2007.

[10] «World Wide Consortium,» [En línea]. Available: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

[11] «Fundamentos, SOAP Versión 1.2 Parte 0:,» World Wide Consortium/es, [En línea]. Available: http://www.w3c.es/Traducciones/es/TR/2003/REC-soap12-part0-20030624/#Example3.

[12] m. conect, «conector OPENERP de magento,» [En línea]. Available: http://www.magentocommerce.com/magento-connect/openerp-bridge.html.

[13] builtwith, «Open Source eCommerce Usage,» 2014. [En línea]. Available: http://trends.builtwith.com/shop/open-source?id=entireInternet#.

[14] Dolibarr, «Funcionalidades Dolibarr,» [En línea]. Available: http://www.dolibarr.es/index.php/erp-dolibarr#funcionalidades.

[15] D. Leffingwell y D. Widrig, «Chapter 14. A Use Case Primer,» de *Managing Software Requirements: A Use Case Approach, Second Edition*, Addison-Wesley Professional, 2003, p. 544.

[16] magentocommerce, Julio 2011. [En línea]. Available:

http://www.magentocommerce.com/wiki/welcome_to_the_magento

_user_s_guide/es/capitulo_1#arquitectura_de_magento.

[17] «REQUIREMENTS, SYSTEM,» magento.com, [En línea]. Available: http://magento.com/resources/system-requirements.

[18] P. GRAND, «Dolibarr The Book 3.6,» Dolistore, 2014, p. 114.

[19] Dolibarr, «Documentacion Triggers Dolibarr,» [En línea]. Available: http://wiki.dolibarr.org/index.php/Triggers-acciones.

[20] B. Delvaux y N. Ferdous, Magento 1.8 Development Cookbook, Packt Publishing, 2014, p. 274.

[21] magentocommerce, «Module Creator,» [En línea]. Available:

http://goo.gl/rbiQDI.

[22] «Estudio sobre Comercio Electrónico 2013,» Observatorio Nacional de las Telecomunicaciones y de la Sociedad de la Información, 2013. [En línea]. Available: http://www.ontsi.red.es/ontsi/sites/default/files/informe_ecomm_2013.pdf.

[23] I.-S. d. prensa, «La analítica de datos o "la nueva materia prima",» [En línea]. Available: http://www-03.ibm.com/press/es/es/pressrelease/43243.wss.

[24] «Black Friday Results 2013,» Hub, IBM-Digital Analytics Benchmark, [En línea]. Available: http://www-01.ibm.com/software/marketing-solutions/benchmark-reports/black-friday-2013.html.

[25] «Real Decreto-ley 13/2012, de 30 de marzo,» [En línea]. Available: http://www.boe.es/boe/dias/2012/03/31/pdfs/BOE-A-2012-4442.pdf.

[26] J. d. Estado, «Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.,» 14 12 1999. [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750. [Último acceso: 2000].

[27] «Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.,» Jefatura del Estado, 14 12 1999. [En línea]. Available: http://www.boe.es/diario_boe/txt.php?id=BOE-A-2014-4950.

GLOSARIO

CRM: Aplicación empresarial, pensada principalmente para el seguimiento de los clientes de una empresa, teniendo almacenado en el CRM toda la información relativa a un cliente.

Meta información: Este tipo de información se usa en las comunicaciones para enviar datos adicionales al mensaje.

BBDD: Son las siglas de Bases de datos.

TPV: Terminal punto de venta, es el dispositivo que tienen en los establecimientos comerciales para la realización del cobro y gestión de los pedidos. En él se pueden imprimir tickets y facturas

Firewall: En español, cortafuegos, es un *software* que controla las conexiones que nuestro ordenador realiza a través de Internet, dando seguridad para evitar conexiones no seguras.

W3C: El World Wide Consortium es un consorcio que redacta recomendaciones para las tecnologías de Internet.

Interfaz de programación de aplicaciones: (IPA) o API (del inglés Application Programming Interface). Es una interfaz que contiene un conjunto de funciones que permite a las aplicaciones o programadores acceder a servicios de una aplicación.

ANEXOS

Anexo 1: SOA y los Servicios Web

En primer lugar decir que SOA no se debe confundir con SOAP. SOA (Arquitectura Orientada a Servicios) es un paradigma de arquitectura para diseñar sistemas distribuidos. En el que las funciones se encuentran definidas como servicios independientes con interfaces invocables. Mientras que SOAP define como deben comunicarse dos programas en distintas máquinas, mediante el intercambio de datos XML.

Anexo 2: Ejemplo Mensaje SOAP

```
<?xmlversion='1.0' ?>
<env:Envelopexmlns:env="http://www.w3.org/2003/05/soap-envelope">

<env:Header>
      <m:reserva xmlns:m="http://empresaviajes.ejemplo.org/reserva"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
            <m:referencia>
            uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d
            </m:referencia>
            <m:fechaYHora>2001-11-29T13:20:00.000-05:00</m:fechaYHora>
      </m:reserva>
      <n:pasajero xmlns:n="http://miempresa.ejemplo.com/empleados"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
            <n:nombre>Pepe Ejemplo</n:nombre>
      </n:pasajero>
</env:Header>

<env:Body>
      <p:itinerario
            xmlns:p="http://empresaviajes.ejemplo.org/reserva/viaje">
            <p:ida>
                  <p:salida>Nueva York</p:salida>
                  <p:llegada>Los Angeles</p:llegada>
                  <p:fechaSalida>2001-12-14</p:fechasalida>
                  <p:horaSalida>última hora de la tarde</p:horaSalida>
                  <p:preferenciaAsiento>pasillo</p:preferenciaAsiento>
            </p:ida>
            <p:vuelta>
                  <p:salida>Los Angeles</p:salida>
                  <p:llegada>Nueva York</p:llegada>
                  <p:fechaSalida>2001-12-20</p:fechaSalida>
                  <p:horaSalida>media-mañana</p:horaSalida>
                  <p:preferenciaAsiento/>
            </p:vuelta>
      </p:itinerario>
      <q:alojamiento
      xmlns:q="http://empresaviajes.example.org/reserva/hoteles">
            <q:preferencia>ninguna</q:preferencia>
```

```
      </q:alojamiento>
</env:Body>
```

```
<?xmlversion='1.0' ?>
<env:Envelopexmlns:env="http://www.w3.org/2003/05/soap-envelope">

<env:Header>
      <m:reserva xmlns:m="http://empresaviajes.ejemplo.org/reserva"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
            <m:referencia>
            uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d
            </m:referencia>
            <m:fechaYHora>2001-11-29T13:20:00.000-05:00</m:fechaYHora>
      </m:reserva>
      <n:pasajero xmlns:n="http://miempresa.ejemplo.com/empleados"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
            <n:nombre>Pepe Ejemplo</n:nombre>
      </n:pasajero>
</env:Header>

<env:Body>
<p:itinerario
      xmlns:p="http://empresaviajes.ejemplo.org/reserva/viaje">
      <p:ida>
            <p:salida>Nueva York</p:salida>
            <p:llegada>Los Angeles</p:llegada>
            <p:fechaSalida>2001-12-14</p:fechasalida>
            <p:horaSalida>última hora de la tarde</p:horaSalida>
            <p:preferenciaAsiento>pasillo</p:preferenciaAsiento>
      </p:ida>
      <p:vuelta>
            <p:salida>Los Angeles</p:salida>
            <p:llegada>Nueva York</p:llegada>
            <p:fechaSalida>2001-12-20</p:fechaSalida>
            <p:horaSalida>media-mañana</p:horaSalida>
            <p:preferenciaAsiento/>
      </p:vuelta>
</p:itinerario>
      <q:alojamiento
      xmlns:q="http://empresaviajes.example.org/reserva/hoteles">
            <q:preferencia>ninguna</q:preferencia>
      </q:alojamiento>
</env:Body>

</env:Envelope>
```

Anexo 3: WSDL

Lenguaje de Descripción de Servicios Web, (Web Services Description Language) es una interfaz basada en xml usada para describir las funcionalidades ofrecidas por un Servicio Web. Además se habla de WSDL para referirse al archivo XML que representa una lectura legible para otras aplicaciones, el cual contiene los métodos que contiene un Servicio Web, sus parámetros de entrada y sus parámetros de salida.

La actual versión de WSDL es WSDL 2.0, donde curiosamente han cambiado las siglas. Ahora se llama: Servicio Webs Definition Language. Se puede ampliar más en el siguiente enlace[17].

Tipos de Datos básicos dentro del WSDL:

- **<types>** Tipo de dato usado en el mensaje, se describe cada uno de los datos y el tipo que tiene asociado.

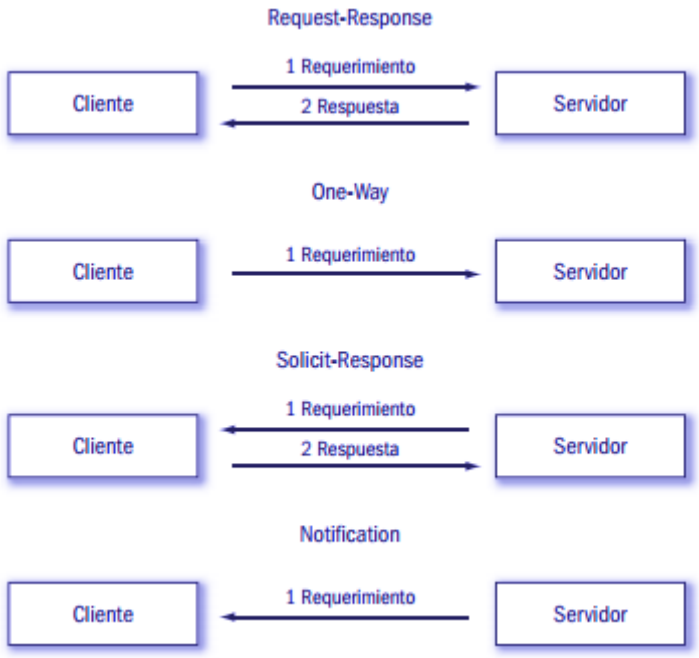
```
<types>
   <schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace
=" urn:Magento">

   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"

   schemaLocation="http://schemas.xmlsoap.org/soap/encoding/"/>
         <complexType name="ArrayOfInt ">...</complexType>
         <complexType name="catalogProductEntity">...</complexType>
         ...
         ...
</types>
```

- **<message>** Se definen los mensajes de entra y salida entre servidor y cliente.

```
   <message name="catalogProductListRequest">
         <part name="sessionId" type="xsd:string"/>
         <part name="filters" type="typens:filters"/>
         <part name="storeView" type="xsd:string"/>
   </message>

   <message name="catalogProductListResponse">
         <part name="storeView" type="typens:catalogProductArray"/>
   </message>
```

- **<PortTypes>** Define el tipo de mensaje a intercambiar entre ambas partes. Hay 4 tipos de mensajes: Request-Response, One-way, Solicit-Response y Notification.

[17] http://www.w3.org/TR/wsdl20/




```
<portType name="Mage_Api_Model_Server_V2_HandlerPortType">
   <operation name="endSession">...</operation>
```

- **<binding>** Se detalla el protocolo que se usará.

```
<binding name="Mage_Api_Model_Server_V2_HandlerBinding"
type="typens:Mage_Api_Model_Server_V2_HandlerPortType">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/
soap/http"/>
   <operation name="endSession">...</operation>
...
...
</binding>
```

- **<service>** Es la lista de puertos y la dirección asociada a estos.

```
<service name="MagentoService">
   <port name="Mage_Api_Model_Server_V2_HandlerPort"binding="typens
:Mage_Api_Model_S   erver_V2_HandlerBinding">
   <soap:address location="http://188.226.164.163/index.php/api/v2_
soap/index/"/>
   </port>
</service>
```

- **<operation>** Define las acciones SOAP, y el formato de codificación del mensaje.

```
<operation name="endSession">...</operation>
```

Anexo 4: Lista programas ERP de código abierto

ERP Package	Lenguaje Base	Licencia	Otra Información	País del Desarrollador
A1 ERP	Java	Alliance Technologies Open License	ERP para Sector público, Académico, Asistencia sanitaria, Logística A1 ERP	Mundialmente
Adaxa Suite	Java	GPL	ERP integrado basado en Adempiere	Australia/Nueva Zelanda
Adempiere	Java	GPL	Comenzó como una bifurcación de Compiere	España
BlueErp	PHP, MySQL, PostgreSQL	GPL		
Compiere	Java	GPL/Comercial	Adquirido por Consona Corporation en junio del 2010	US
Dolibarr	PHP	GPL		
ERP5	Python, Zope, MySQL	GPL	Basado en un modelo unificado	Brasil, Francia, Alemania, Japón, Senegal
ERPNEXT	Python, JavaScript, MySQL	GPL	ERP para pequeñas y medianas empresas	India
Fedena	Ruby, MySQL	Apache License	ERP para Escuelas/Universidades	India
GNU Enterprise	Python	GPLv3		
HeliumV	Java	AGPL	ERP para pequeñas y medianas empresas	Austria, Alemania
iDempiere	Java	GPL	Bifurcación de Adempiere/Compiere	Mundialmente
JFire	Java	LGPL		
Kuali Foundation	Java			
LedgerSMB	Perl, PostgreSQL	GPL	Comenzó como una bifurcación de SQL-Ledger en 2006	Mundialmente
OFBiz	Apache, Java	Apache License 2.0	ERP para pequeñas y medianas empresas	
Openbravo	Java	Openbravo Public License (OBPL), un software libre cuya licencia está basada en el Mozilla Public License (MPL)		España
OpenERP	Python, PostgreSQL	AGPLv3, OpenERP Public License	Conocido anteriormente como Tiny ERP	Bélgica, India, US
Opentaps	Java	AGPLv3	Puede ejecutarse en Amazon EC2 cloud [1]. Basado en Apache OFBiz 10.04 y Tomcat 6.0.26	Mundialmente
Postbooks	C++, JavaScript, PostgreSQL	CPAL	Producido por XTuple, usa las librerías Qt	

Tabla 42: ERP de código abierto

Anexo 5: Funcionalidades Dolibarr

Gestión de Usuario	• Gestión de usuarios del sistema. • Personalización de temas por usuario. • Individualización del escritorio (Dashboard). • Gestión de usuario externos (perfecto para dar accesos a clientes y ahorrar tiempo de gestión). • Gestión de grupos. • Granularidad de permisos por grupos y usuarios. • Conectividad de usuarios a través de LDAP. • Número ilimitado de usuarios. • Varios administradores.
Gestión de productos/ Servicios	• Gestión completa de productos y servicios. • Gestión de almacenes y stock de productos. • Establecimiento de distintos niveles de precios. • Estadística de ventas por productos. • Categorización de productos y servicios. • Gestión de envíos y transportes.
Gestión comercial	• Gestión completa de presupuestos. • Gestión completa de pedidos a clientes y proveedores. • Generación de notas de entregas o albaranes. • Flujo de trabajo entre pedidos y presupuestos. • Estadísticas de presupuestos y pedidos.
Gestión financiera	• Gestión de facturas a clientes y proveedores. • Gestión de impuestos y cargas sociales. • Informes de resultados. • Gestión de cuentas bancarios y otras. • Gestión de domiciliaciones bancarias. • Gestión automática en los pagos de facturas de las domiciliaciones. • Flujo de trabajo con la parte comercial. • Estadísticas financieras.
Gestión de proyectos	• Organización de proyectos. • Creación y seguimiento de tareas asignadas a un proyecto. • Asignaciones de proyectos y tareas. • Seguimiento de la progresión. • Sencillo diagrama de Gantt.
Gestión de terceros	• Gestión de clientes, clientes potenciales (prospectos), proveedores y contactos. • Gestión de clientes, clientes potenciales (prospectos), proveedores y contactos. • Gestión de categorías de terceros para un mejor control. • Gestión de descuentos de terceros. • Vinculación de la agenda y eventos (creación de presupuestos, pedidos, facturas, envíos de emails, etc.).

Seguimiento de terceros CRM	• CRM con todas las entidades: presupuestos, intervenciones, pedidos, facturas, etc. • Registro automático de acciones y eventos en la agenda automáticamente. • Seguimiento de contactos realizados con terceros. • Anotaciones privadas.
Terminal punto de venta (TPV)	• Sistema básico de terminal punto de venta. • Ventas de mostrador con código de barras. • Integrado con el control de inventario y con la gestión financiera (creación de facturas).
Gestión documental	• Sistema básico de almacenamiento de documentos. • Gestión documental integrada en las entidades: pedidos, facturas, terceros, etc. • Recomendado para comerciales que viajan con periodicidad.
Gestión de contratos	• Avisos en el dash booard con umbral definido. • Facturación de contratos con un solo click. • Seguimiento de los productos y servicios de venta periódica y repetitiva. • Ahorra tiempo y costos.
Gestión de intervenciones	• Creación y gestión de hojas de intervenciones. • Controles de tiempo. • Gestión de las tareas. • Creación de facturas a partir de las intervenciones con un solo click.
Gestión de registro y agenda	• Agenda para la gestión de eventos. • Compartición de la agenda con otros usuarios. • Integración con google CAL (necesario módulo). • Registro automático de todas las acciones del usuario (creación de presupuestos, etc.). • Listado de acciones desde el calendario. • Creación de PDF de acciones.
Administración del sistema	• Completa gestión de la parametrización del sistema. • Controles de tiempo. • Gestión de usuarios. • Gestión de diccionarios. • Gestión de límites y precisiones del sistema. • Control de la seguridad del sistema. • Gestión de los paneles. • Gestión de *backups*. • Gestión de menús. • Gestión de SMS. • Gestión del entorno. • Gestión de alertas. • Configuración a medida y flexible. • Control de flujos de trabajo.

Miembros	• Gestión de miembros (indicado para asociaciones, clubs, etc. empresas que cuentan con socios) y sus cuotas. • Gestión de subvenciones. • Gestión de bookmarks. • Posibilidad de multiempresa (a través del módulo multicompany). • Multitud de módulos externos que dotan a Dolibarr de más funcionalidades.
Y más…	• Gestión en PDF de los presupuestos, pedidos, facturas, notas de entrega, intervenciones, etc. y envío por correo desde la misma aplicación, sin salir del programa. • Gestión de importación y exportación de datos a CSV, TSV y XLS. • Gestión de EMAILING (newsletters). • Gestión de expediciones.

Tabla 43: Funcionalidades Dolibarr. Fuente: Dolibarr

Anexo 6: Configuración servidor LAMP

Suponiendo que se tiene un servidor con Ubuntu 12.04 ya instalado (común en todas las empresas de alojamiento *Web*) se procede a detallar la configuración básica en los siguientes pasos:

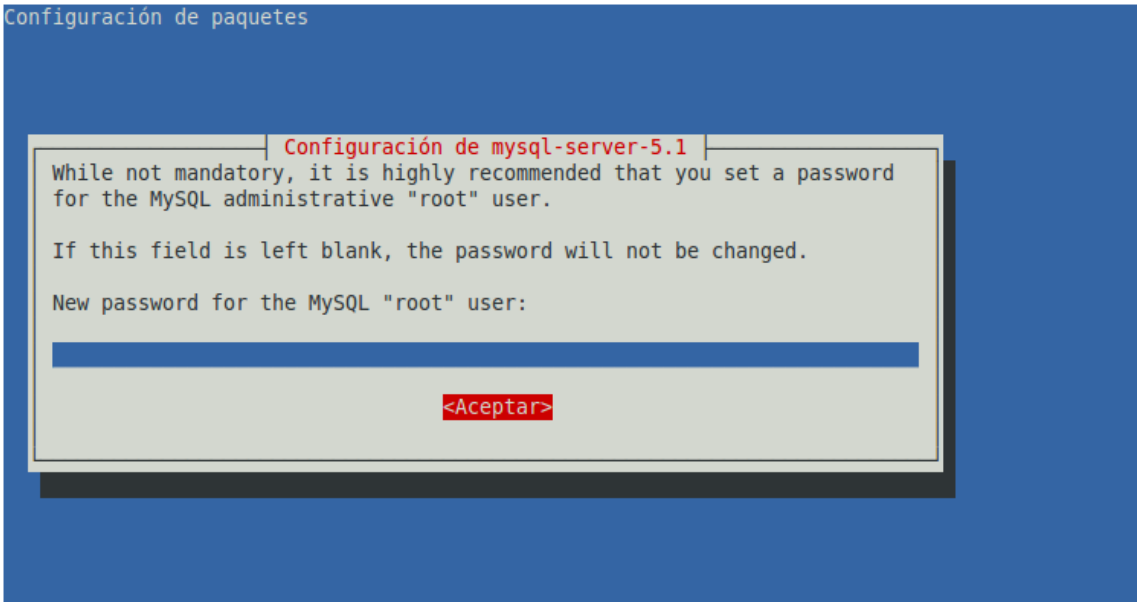
Instalación de apache:

```
sudo aptget install apache2
```

Instalación de MySql:

```
sudo aptget install mysql-server
```

Durante la instalación de MySQL es necesario escribir la contraseña de administrador (de MySQL, no confundir con el root de Ubuntu):



Instalación de PHP:

```
sudo aptget install php5 libapache2-mod-php5 php5-mysql php5-mcrypt
```

Reiniciar apache para que se graben los cambios en el servidor (IMPRESCINDIBLE).

```
sudo /etc/init.d/apache2 restart
```

Instalación de Phpmyadmin para el manejo de bases de datos:

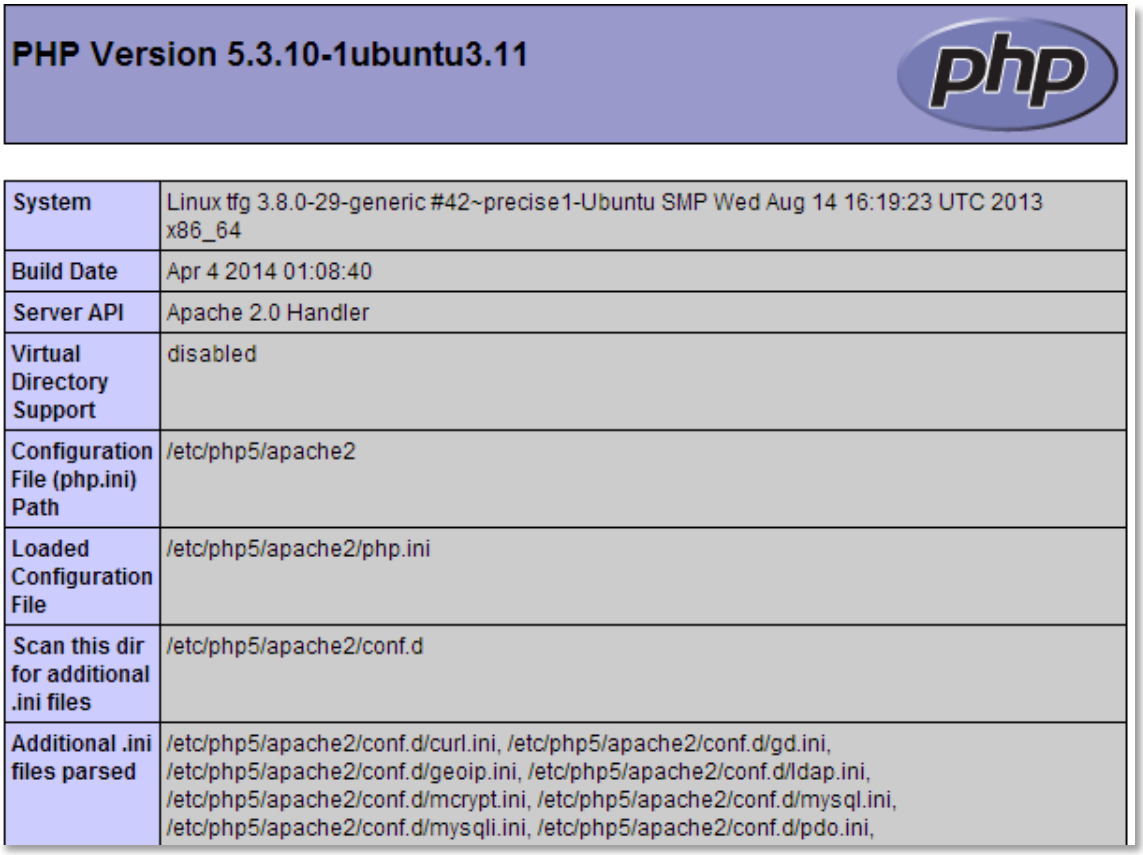
```
sudo aptget install phpmyadmin
```

Con esto se tiene un archivo básico de php ya en funcionamiento por ejemplo phpinfo.php con el código:

```
<?php
phpinfo();
?>
```

Y se ve que funciona correctamente si se accede a:

http://95.85.21.243/phpinfo.php



PHP Version 5.3.10-1ubuntu3.11

System	Linux tfg 3.8.0-29-generic #42~precise1-Ubuntu SMP Wed Aug 14 16:19:23 UTC 2013 x86_64
Build Date	Apr 4 2014 01:08:40
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/curl.ini, /etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/geoip.ini, /etc/php5/apache2/conf.d/ldap.ini, /etc/php5/apache2/conf.d/mcrypt.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini,

Fuente: DigitalOcean - How to Install Linux Apache Mysql PHP stack on Ubuntu.

https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu

Anexo 7: Instalación Magento

Se describe a continuación la manera de instalar Magento en un servidor. Primero se prepara el entorno para Magento, después se procede a su descarga e instalación.

En primer lugar se requieren unas modificaciones en el archivo "php.ini" hay que editar:

```
nano /etc/php5/apache2/php.ini
```

Y cambiar la memoria del servidor mínima de 128 a 512 (o superior si se puede).

```
memory_limit = 512M
```

Se requieren librerías adicionales de php para que pueda funcionar la plataforma Magento:

```
sudo apt-get install libcurl3 php5-curl php5-gd php5-mcrypt
```

Magento en ciertas situaciones requiere enviar emails (si no se quiere configurar un servidor SMTP) por lo que es necesario instalar:

```
sudo apt-get update
sudo apt-get install sendmail
sudo sendmailconfig
```

Se reinicia apache:

```
sudo service apache2 restart
```

Magento necesita una base de datos para funcionar, en el Anexo anterior ya se muestra como instalar MySQL, por lo que tan solo es necesario crear una nueva BBDD para Magento:

```
create database magento;
```

Ya se han instalado y configurado todas las herramientas que necesita Magento, ahora se pasa a su descarga e instalación. En primer lugar se consigue la versión más reciente de Magento, en este caso la 1.9.

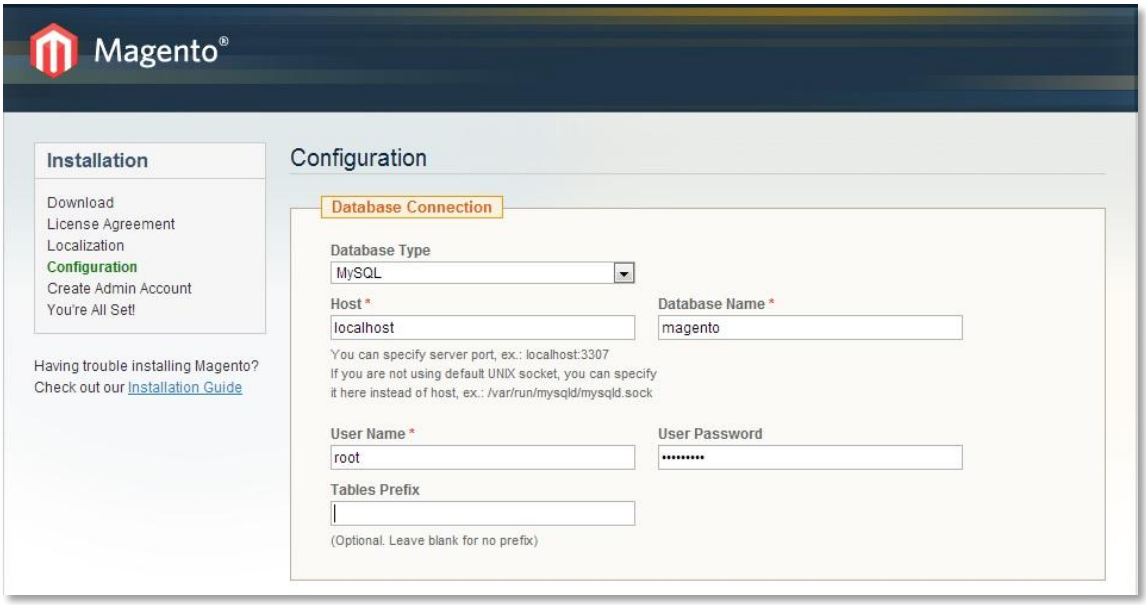
```
wget http://www.magentocommerce.com/downloads/assets/1.9.0.1/magento-1.9.0.1.tar.gz
```

Se descomprime y configura correspondientemente la carpeta con unos permisos para que apache pueda usar esos directorios:

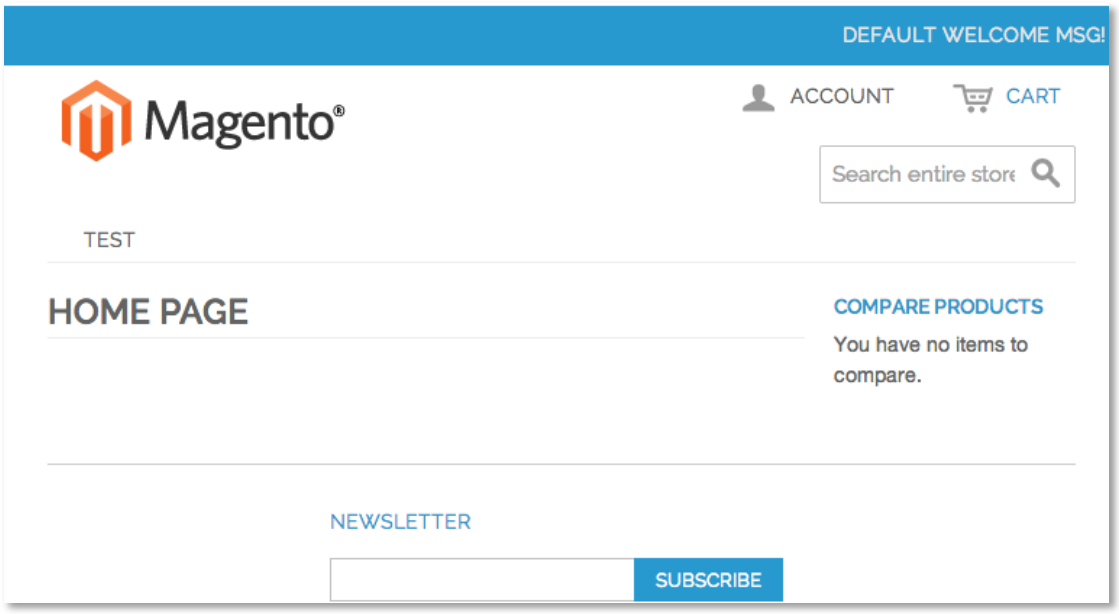
```
tar -zxvf magento-1.9.0.1.tar.gz

chmod -R o+w media var

chmod o+w app/etc
```

A partir de aquí es necesario pasar al navegador, acceder a nuestra IP o dominio y realizar la instalación desde el asistente en el navegador *Web*.

- Se elige la localización de la tienda: Zona horaria y moneda.
- El tipo de BBDD: En este caso MySql
- El nombre de la BBDD: según lo creado en el punto anterior "magento" y la contraseña.



Y con esto ya se tiene Magento instalado:



Fuente: http://www.magentocommerce.com/wiki/1_-_installation_and_configuration/installing_magento_via_shell_ssh#installing_magento_via_ssh

La guía de usuario de Magento, se puede encontrar en el siguiente enlace:

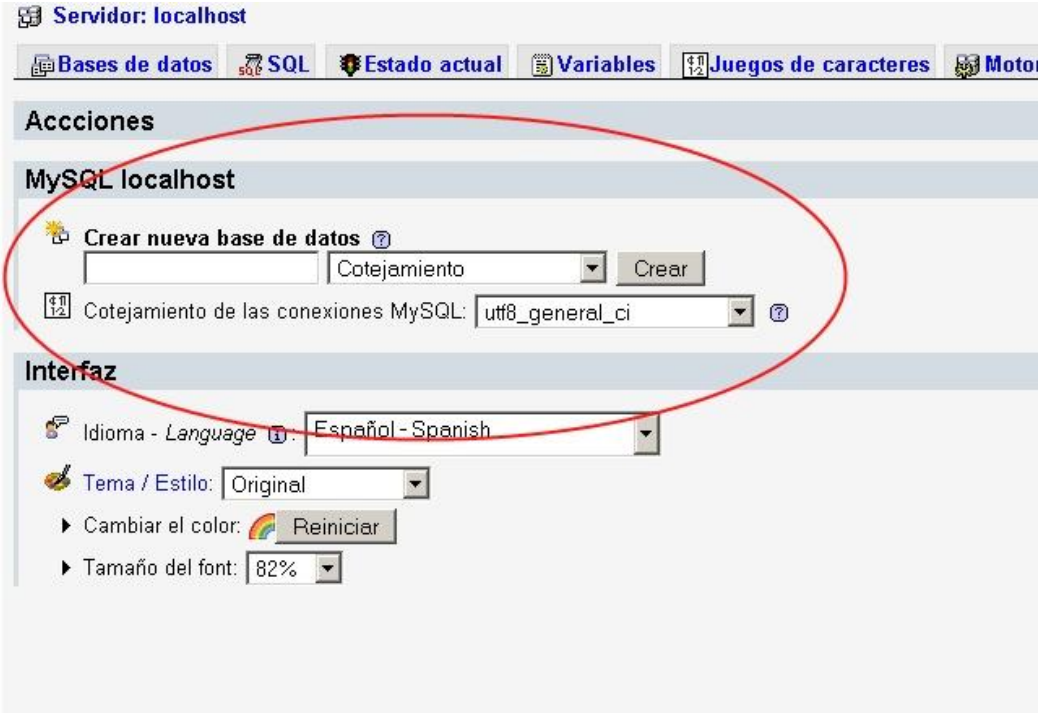
http://www.on4u.es/docs/doc/ManualMagento.pdf

Anexo 8: Instalación Dolibarr

Descargar la última versión de Dolibarr:

http://www.dolibarr.org/downloads
Subirla al servidor DESCOMPRIMIDA vía FTP.

Crear la BBDD en phpmyadmin:



Después es necesaria la ejecución del siguiente comando:

```
sudo dpkg -i dolibarr_x.y.z-w.w_all.deb
sudo apt-get install -f
```

Donde x.y.z es la versión actual de Dolibarr por ejemplo: dolibarr_3.5.3-1.1_all.deb

Una vez instalado se puede acceder a:

http://95.85.21.243/dolibarr/

Y se ve Dolibarr funcionando:

La guía de usuario de Dolibarr, se puede encontrar en el siguiente enlace: www.dolistore.com/attachment.php?id_attachment=101